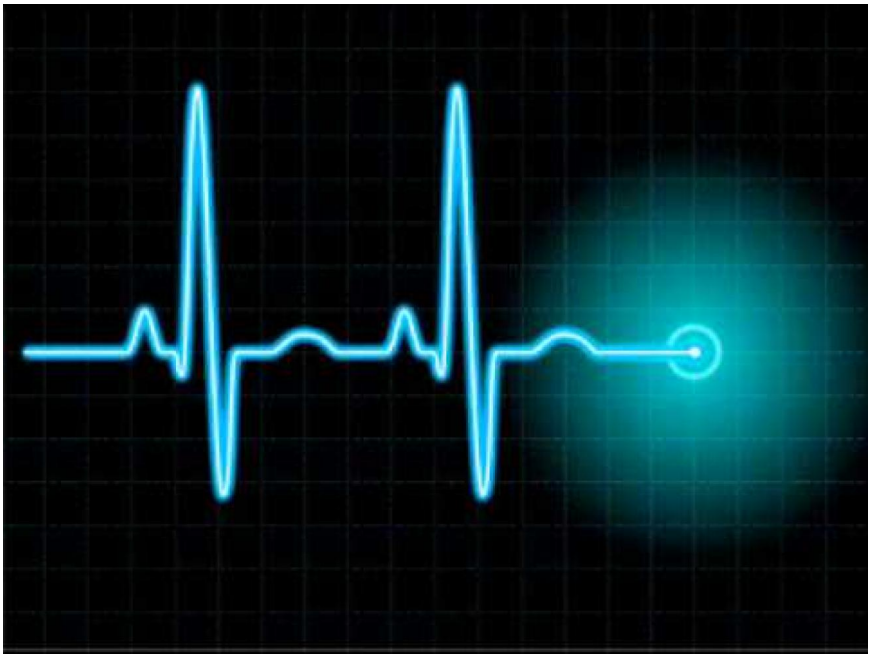

Evaluating the Health of Open Source Components

2013

Master Thesis
24/06/13

Ivan Zaytsev (3775674)
i.zaytsev@students.uu.nl

MSc Business Informatics
Institute of Information and Computer Science
Utrecht University

Utrecht University

IBM

Master thesis ***Dec. 2012 – Jun. 2013***

This document is a master thesis written within the information and organization research group of the Institute of Information and Computer Science at Utrecht University. The research was conducted in cooperation with IBM Belgium.

PROJECT SUPERVISORS

First supervisor:
dr. S.R.L. Jansen
Utrecht University

Second supervisor:
prof. dr. S. Brinkkemper
Utrecht University

Academic relations coordinator
PhD Hans Van Mingroot
IBM BeLux:

FACILITATING INSTITUTIONS

UNIVERSITY

Master Business Informatics
Department of Information Science
Utrecht University

RESEARCH PARTNER

IBM Belgium/Luxembourg
Avenue du Bourget / Bourgetlaan 42,
1130 Brussels

GRADUATE STUDENT

Name	Ivan Zaytsev
Student-No	3775674
E-mail	i.zaytsev@students.uu.nl
Master Program	MSc Business Informatics
Thesis duration	1. Dec 2012 – 24. Jun 2013
Thesis title	Evaluating the Health of Open Source Components

Abstract

Implementing open source components into commercial applications can add new and innovative functionality to commercial software products. Integrating open source enables professional software developers to optimize development efforts and free up time and resources, which can be used to add new functionality to existing applications. However, contrary to often discussed legal limitations of open source, the aspect of community vitality is rarely addressed. An unforeseen decline in health of a supplying community can lead to software bugs that cannot be fixed, a decrease in overall software quality and large expenses, caused by transition costs to an alternative software component. Software product managers who want to benefit from utilizing open source in their products must take these and many other related risks into account.

In order to support software product managers and enable them to increasingly rely on open source, this research systematically analyzes the vitality of open source communities and introduces a structured method for open source vitality analysis. Built on contemporary scientific literature and expert interviews conducted at IBM, the Open Source Health Analysis Method is introduced. The method is modular and specifically designed to be adjustable to situational requirements of varying communities and project characteristics. In order to allow product managers to respond to changes in community health, an open source interaction model presents a number of activities that can be taken in order to stimulate community growth and vitality. The introduced models enable software product managers to confidently implement open source and add new and innovative features to their managed software products.

Table of Content

Table of Figures

Table of Tables

1. Introduction

Contemporary, large software products are frequently developed by an entire ecosystem of organizations and open source communities (Jansen, Finkelstein, & Brinkkemper, 2009). Such software products base their code on deliverables produced in so called software ecosystems, which span beyond the influence of a single organization (Bosch, 2009). Hereby, open source communities often provide innovative, user-demand driven software, free of charge (Von Hippel, 2001).

Implementing open source components into commercial applications has many advantages for software developers, as they can reduce development costs, improve an application's performance and add functionality without the need to invest into according in-house capabilities (Bessen, 2001). Development resources, freed up by open source integration, can be invested in strengthening a software product's core capabilities and improving its market competitiveness (Hawkins, 2004).

However, due to a different development approach then with commercial software applications, open source integration also bears a number of risks for software firms as they must guarantee service level fulfillment for each application throughout its life cycle.

1.01 Problem Definition

For developing and maintaining a software product that depends on open source code, product managers must be able to assess the health of the communities their products depend on. A lack of understanding of a software component's vitality bears not quantifiable risks for the software firm, as well as for customers relying on the product in question. However, such an assessment can be particularly difficult, as open source communities are loosely structured and use collaboration methods that differ greatly from those in commercial software development (Crowston & Howison, 2005). Additionally, an open source communities' expertise is not centrally structured, less tangible than with commercial products, and bears no obligations for formal technical support.

Based on the arguments above, the problem definition of this thesis project is as follows:

> "Product Managers of commercial software applications have no systematic approach for assessing the health and vitality of the open source components that their products depend on."

Without a systematic evaluation of community health, software products with application lifecycles spanning across many years are in danger of becoming dependent on a dying community that no longer updates or maintains its code base and has no users that could provide technical support. In such a situation a software company can face unforeseen complexities in update and patch development, which can result in a considerable increase in support costs or legal expenses. Additional complications can stem from transaction costs, created by the necessity to switch to an alternative software component. Even if the replacement is another open source solution, transitioning requires unscheduled time and resources for implementation. Thus, development resources that were originally freed up by the utilization of the original open source component are no longer available to new development activities.

1.02 Objective and Problem Statement

The purpose of this research is to develop a method for software product managers (SPMs) that will allow them to assess the health and vitality of open source communities and enable them to respond to a decline in vitality in due time. Focusing on product managers of commercial products (Van De Weerd, Brinkkemper, Nieuwenhuis, Versendaal, & Bijlsma, 2006), the aim is to clearly define relevant vitality indicators of open source communities and devise a method for measuring current and past community health. Such an understanding can support SPMs in making strategic decisions in regard to component selection, internal resource allocation and the product's marketing strategy.

Based on the arguments above, the formal research question of this thesis is:

> "How can Software Product Managers systematically and strategically evaluate the health and implementation of open source components within their commercial software products?"

In order to address this objective, relevant vitality indicators are being analyzed and current evaluation practices at a case company studied. The analysis and the research conclusions are utilized to lay out options for responding to changes in community health.

The objective of the overall research is defined as follows:

> "Enable Software Product Managers to measure the health status of open source communities, respond to health changes, contribute to the overall quality of commercial software products and provide scientific insights into relevant vitality indicators for community health."

1.03 Research Questions

Extending the introduced research model, the main research question and respective sub-questions are formulated:

How can product managers of software products, utilizing open source components, assess the health of open source communities that their product depends on?

1. Which are the relevant vitality indicators for community health?
2. Is community health measurable and comparable by means of an evaluation method?
3. Which are suitable analysis techniques to demonstrate changes in community activities over time?
4. Which measures can software product managers take to respond to changes in the health of open source communities that their product depends on?

All presented research questions will be directly addressed by dedicated thesis deliverables throughout the following chapters (see Section 2.03). Additionally, all questions and sub-questions will be matched with corresponding thesis findings within the conclusion of this document (see section 0), thus aggregating the final research implications of this thesis.

1.04 Terminology and Scope

A definition of relevant terms for this thesis project can be found below. All terms are based on recent scientific literature and are adjusted to the scope of the thesis.

Open Source Community / Open Source Component

Within this thesis the terms software community and software component are used interchangeably. Hereby, the term open source community (OSC) refers to a community of individuals contributing to a software project with the intention of making the source code available to everyone to inspect, change, download, and explore as they wish. The focus of this work lies on OSCs with a licensing model that permits for unrestricted commercial utilization, such as software published under the MIT[1] license. No distinction between conventional licenses or dual licensing models (Välimäki, 2003) is made.

Community Health

OSC health refers to the activity and vitality of a given community (K J Stewart & Ammeter, 2002) at a certain time and can be measured in variables, such as the frequency of major or minor software releases, the average number of unresolved bugs, the number of active developers, the popularity among developers outside of the OSC, as well as many other contributing factors or variable constructs. An additional focus of this thesis lies on long-term component stability and aims at assessing development trends within OSCs.

Commercial Software Product

All commercial software products, analyzed within this thesis project, include or rely on OSCs for their functionality. No distinction between their licensing models or a differentiation between individually developed products or software product lines (Northrop & Jones, 2004) is made. Within this thesis, the applied definition of the term is deliberately kept broad and refers to almost any software application that is for sale.

Software Ecosystem

Often seen as a consecutive development to software product lines (Bosch, 2009), this research follows the definition of Jansen, Finkelstein and Brinkkemper (2009):

"…a set of businesses functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently under- pinned by a common technological platform or market and operate through the exchange of information, resources and artifacts."

[1] http://opensource.org/licenses/MIT

Software Product Manager

The software product manager (SPM) is a role, filled by either an individual or a team, responsible for the management activities related to development, marketing and sales of commercial software products. The responsibilities of a SPM can include: product requirements, release definition, product release lifecycles management, as well as the coordination of development, marketing and sales activities (Brinkkemper, Ebert, & Versendaal, 2006)

Open Source Community Health Analysis Method

The OSC health analysis method rests upon method engineering principles, such as those introduced by Brinkkemper (1996), and is a structured approach, which can be used by software product managers to assess the health of OSCs. Such a method is composed of a set of steps and according deliverables, which SPMs can perform in order to systematically aggregate relevant data for OSC health evaluation.

1.05 Scientific relevance

The following section describes the scientific contribution, as well as the practical value of this thesis project.

Scientific Contribution

The scientific field of Software Ecosystems is relatively new and dates back to as late as 2005. Jansen, Finkelstein and Brinkkemper (2009) state that "SECOs introduce many new research challenges on both a technical and a business level" This thesis contributes to the SECO domain in two ways. First, it enhances the understanding of SECO structures and collaborations within ecosystems with, both, commercial and open source software components. Second, it provides a qualitative and data-driven insight into open source community health, which is currently a relatively novel domain with few contributions to it. Furthermore, the research directly affects the existing body of knowledge on systematic open source vitality assessment, community interaction and vitality stimuli.

Practical Value

Focusing on business value and practical applicability the purpose of this thesis is to enable software product managers to gain a better control and understanding of all relevant components of their respective software products. By applying a structured, analytical approach to open source software within their products, PMs will be able to continuously analyze and respond to changes in OSC health, which otherwise might affect their product negatively. Furthermore, this research provides software firms with the opportunity to actively use open source as a means of marketing for their commercial products. Evidently healthy and mature open source components can be seen as proof of quality of the commercial product and illustrate that it is fully focused on its core competencies. Such an approach enables software firms to focus on core development activities, while ensuring a broader set of features to their clients.

1.06 Thesis Content

Within the following chapters, the presented research questions will be systematically addressed and answered. Hereby the thesis content is structured as follows:

Chapter	Content
Research Method	The applied research model, the operationalization approach, research deliverables, design principles and result evaluation.
Theoretical background	A general introduction into the software ecosystem domain and software product management, as well as a systematic literature review on open source vitality.
Understanding of Open Source Communities	Overview of open source participators, a community interaction model, community health indicators and an outline of corporate participation in open source.
Industry insights in Open Source practices	A practical insight into Open Source related practices at IBM with a specific focus on internal open source processes.
OSC Health Analysis Method	Introduction of a method fragment pool and a fragment base for vitality analysis, as well as fragment assembly, calibration and selection principles. Furthermore, the chapter includes a specific example of method application for community assessment.
Method validation	Use case driven validation of method applicability and method usability by open source experts.
Conclusions	Final research results, existing limitations, recommendation for future research, as well as an assessment of the practical value of the presented method.
Bibliography	Cited scientific literature throughout the thesis.
Appendices	Deliverables, too large or unsuitable to be placed within the main body of the research document.

Table 1 – Thesis Content Overview

2. Research Method

The following section introduces the research method of this thesis. The method is segmented into the research model, the operationalization approach, the main research deliverables, result validation and research design principles.

2.01 Research Model

Following up on the introduced research questions, the research model, depicted in Figure 1, consists of three main research phases, as well as a preliminary phase aimed at defining the research scope and research approach. Concepts of method engineering and the aggregation of method fragments (Van De Weerd, 2008) are used to illustrate the necessary steps towards creating the OSC health analysis method and answering the corresponding research questions. Utilizing a so called process deliverable diagram (PDD), the left side of the model depicts the meta-process model which is comprised of the detailed activity flow. The right side illustrates the meta-deliverable model comprised of the associated concepts linked to the respective origin activities.

The following sub-sections contain a description of the intent of each research phase.

2.01.1 Preliminary phase

Prior to operationalization, preliminary steps towards the thesis project deal with formulating the research approach and defining the domain and the research scope. The utilized approach consists of research activities and according deliverables, structured in three consecutive phases. The individual phases are designed to answer the sub-research questions (see Table 2) and lead to the delivery of the final OSC health analysis method.

2.01.2 Method foundation

This qualitative research phase focuses on aggregating scientific literature and creating an overview of current practices for OSC integration and management within the case company. A systematic literature review is used to discover and analyze contemporary literature on OSC and software ecosystem vitality. The empirical perspective is covered by means of a set of interviews with open source experts, conducted at IBM Benelux. Consequently, the results of both steps are being processed, categorized and aggregated into a list of vitality indicators for OSC. Furthermore, an OSC interaction model, including corporate stakeholders, is designed. The model highlights relevant OSC actors, their respective roles within the community, as well as the discovered incentives for OSC participation.

2.01.3 Method design

Based on the categorized vitality indicators, method fragments for health analysis are designed and aggregated into a fragment pool. Each fragment contains a set of activities and corresponding deliverables, which can be used individually, or in combination with other fragments, to analyze a particular aspect of OSC health. Furthermore, each fragment is associated with a selection rationale that is based on the introduced OSC interaction model. In order to demonstrate the method's practical applicability, a representative OSC is selected and a suitable OSC health analysis method designed. The example illustrates applied principles of fragment selection, method assembly and calibration.

2.01.4 Method validation

The method fragment pool, as well as all individual fragments, underwent a validation of their suitability for OSC vitality analysis. The applied approach was based on 10 semi structured interviews with open source experts at the case company. By presenting the experts with a number of different OSC integration scenarios, the applicability of the devised approach was evaluated and rated.

Specifically, looking at reproducibility of the thesis results, validity was addressed in the following way:

Internal validity

The used research approach addresses internal validity from both, a scientific, as well as an empirical angle. The vitality indicators, included in the method fragment pool, have been identified by at least two scientific publications, open source experts, or by a combination of both sources. Furthermore, the OSC Health Analysis Method is supported by an OSC interaction model, derived from contemporary open source research, as well as an evaluation of current OSC related practices at the case company.

Threats to internal validity are based on the degree of applicability of the method for software product managers and the method's accuracy in regard to predicting OSC vitality. Additionally, the unique characteristic of each OSC and the resulting necessity to adjust situational fragments further complicates validity evaluation. The research method addresses these threats with the aforementioned rigid vitality indicator assessment as well as semi structured interviews with domain experts.

External validity

Due to the nature of situational method engineering and the fact that all individual fragments are based on vitality indicators, derived from scientific research, as well as expertise built on a large number of varying OSCs, the authors believe that the results are generalizable to a broad variety of communities and software product managers.

Alike to the threats to internal validity, external validity is also affected by an uncertainty regarding method applicability for varying types of software projects, as well as generalizability of accuracy in predicted OSC vitality. Both concerns are addressed with the same means as in the case of internal validity.

Reliability

Research reliability is addressed by means of a thoroughly documented systematic literature review and interview guidelines. Thus, all identified vitality indicators, as well as the OSC interaction model can be fully reproduced. Furthermore, the suitability and applicability of all method fragments can be validated based on a fully documented interview protocol.

Looking at method accuracy, research reliability is affected by the applied situational approach to method design. Thus, only the same OSC can be directly compared, while reviewing method accuracy across varying communities becomes more difficult. In such cases, each community must be evaluated while taking its unique characteristics into account, making a direct comparison less accurate.

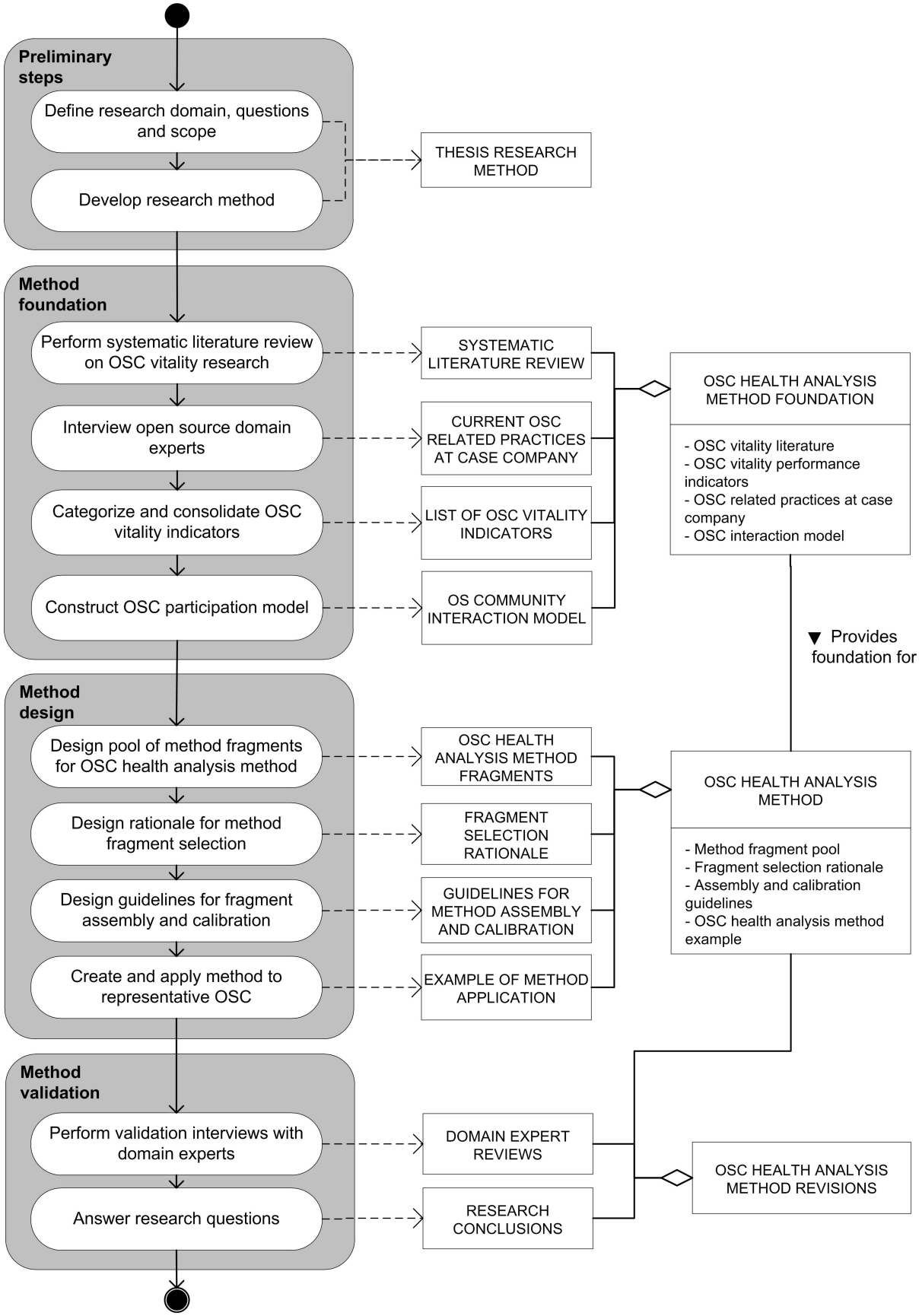

Figure 1 - Process Deliverables Diagram: Research model and research phases

2.01.5 Activities linked to sub research questions

Table 2 links sub research questions to their corresponding deliverables within the research method and explains how each deliverable will contribute to answering them.

Sub research question	Description	contributing deliverables
Q1: Which are the relevant vitality indicators for community health?	Community health performance indicators are derived from existing scientific SECO research and interviews with open source experts at the case company.	• SYSTEMATIC LITERATURE REVIEW • CURRENT OSC RELATED PACTICES AT CASE COMPANY • OSC INTERACTION MODEL • LIST OF OSC VITALITY INDICATORS
Q2: Is community health measurable and comparable by means of an evaluation method?	This question will be addressed with results from the systematic literature review, the constructed community interaction model and the resulting selection rationale for method fragments.	• GUIDELINES FOR METHOD ASSEMBLY AND CALIBRAITON • OS COMMUNITY INTERACTION MODEL • FRAGMENT SELECTION RATIONALE • OSC HEALTH ANALYSIS METHOD REVISIONS
Q3: Which are suitable analysis techniques to demonstrate changes in community activities over time?	Analyzing data trends, over time, this question will be answered with the operationalization of the OSC vitality indicators and additionally validated within the final method design phase.	• LIST OF OSC VITALITY INDICATORS • OSC HEALTH ANALYSIS METHOD FRAGMENTS • GUIDELINES FOR METHOD ASSEMBLY AND SELECTION RATIONALE • EXAMPLE OF METHOD APPLICAITON
Q4: Can the health of an open source community be improved by means of intervention of a commercial party?	This primarily qualitative question will be answered based on results of interviews conducted at the case company, the OSC interaction model and results of the systematic literature review.	• CURRENT OSC RELATED PACTICES AT CASE COMPANY • OS COMMUNITY INTERACTION MODEL • SYSTEMATIC LITERATURE REVIEW

Table 2 - Research Deliverables Matched With sub Research Questions

2.02 Operationalization

The empirical foundation for the OSC Health Analysis method is based on a systematic literature review, a study of current practices at the case company, a developed community interaction model, as well as five expert interviews, conducted at the globally operating case company with extensive open source expertise.

2.02.1 Systematic Literature review

Focusing on the scientific body of knowledge on OSC and SECO health, a systematic literature was performed (see Section 3.03). The operationalized data was structured and grouped by contemporary SECO research domains. The gathered data covers:

- OSC Community Practices
- Community and Project Popularity Assessment
- Commercial Community Sponsorship and Licensing
- OSC Software Quality Assessment

2.02.2 Expert Interviews

The understanding of the case companies' current OSC practices as well as the OSC Health analysis method validation are based on an analysis of internal documentation, as well as ten semi-structured, explorative interviews with IBM employees stating open source as one of their main responsibilities or fields of expertise.

Hereby, the review of the current practices is based on five interviews, in combination with an analysis of internal documentation, while the method validation was performed with a total of ten interviewees. It is furthermore noteworthy that, five out of ten interviewees were selected based on personal recommendations by senior software developers or IT architects, highlighting their expertise in open source.

Interviewee Selection

Based on an internal employee directory, a search has been conducted, specifically looking for employees listing open source, OSCT or OSSS (see Section 5.01.1) as a key responsibility of their function at IBM. As the case company does not have a direct equivalent to the job role of a software project manager, OS expertise has been the primary selection criteria. Furthermore, no preference, to geo-location or business unit has been given as IT experts can often be found across different product brands or service lines within IBM.

Interview Purpose

The main aim of the conducted interviews is divided across two interview rounds, both having a different research purpose.

The intent of the first round was to gain a deep understanding of IBM's current OSC related practices and to contribute to the collection of OSC vitality indicators. In combination with the indicators from the literature review, the resulting list thus covers both, theoretical and practical perspectives on OSC vitality. Furthermore, the explorative nature of the interviews allowed for questions about practical experiences with OSC utilization in commercial products or projects and the experiences, the interviewees had made with open source, within commercial software development environments.

The second interview round served the purpose of validating the devised OSC health Analysis method from the standpoint of open source experts. Hereby, the interviewees were asked to rate the usability of each fragment individually, as well as to assess if method assembly or method calibration were suitable for overall vitality analysis. Additionally, all interviewees were presented with a number of evaluation scenarios (see Appendix B) in which they were supposed to review the method's practical value.

Interviewee process

In order to ensure a systematic and a reproducible interviewing process, all interviewees received the same research participation invites for each round. Furthermore, each interview followed the same set of steps and predefined questions (see Appendix B). The conducted interviews can be categorized as semi structured.

In the case of method validation, all interviewees were read three predefined evaluation scenarios and were given a printed copy of each method fragment that they were encouraged to write on. If beneficial to the research, additional space was given to explore selective topics in greater detail. For verification purposes, all responses were transcribed and read back to the interviewees, immediately following the completion of the interviews. The first interview round, aimed at discovering vitality indicators and current practices took on average 30 minutes. The second round had the purpose of validating the introduced OSC Health Analysis method and took an average of 45 minutes per interview.

2.03 Section Deliverables

Within this section an overview and the descriptions of the main thesis deliverables are introduced:

2.03.1 Systematic literature review

A systematic literature review on the topics of software ecosystems and OSC health has been performed. The review forms the theoretical foundation of this research and contributes to a number of deliverables, such as the list of vitality indicators, the OS interaction model and the method fragment selection rationale.

2.03.2 Open Source Community vitality indicators

A set of interviews with software product managers at the case company, in combination with the aforementioned literature review, provides the necessary inputs to design a list of categorized OSC vitality indicators. The indicators are of both, qualitative and quantitative nature and allow for the design of method fragments that systematically measure OSC health aspects.

2.03.3 Open Source Community Interaction Model

The interaction model illustrates all relevant OSC participants and describes the incentive structures of individual groups of contributors. Furthermore, the model highlights how particular groups of OSC participators influence the motivation of other OSC players and allows for drawing conclusions regarding corporate participation in open source communities.

2.03.4 Pool of OSC health analysis method fragments and selection rationale

Following method engineering principles for situational method design, the method fragment pool represents a collection of fragments that can be used individually or in combination by software product managers for OSC vitality assessment. Each fragment is labeled with three selection indicators that support SPMs assessment of fragment suitability.

2.03.5 Example Utilization of the OSC Health Analysis Method

The final deliverable presents a practical example of method assembly and application that illustrates all necessary steps in order to create and calibrate an OSC vitality assessment. Specifically, the example highlights the necessary design and calibration steps for creating performance indices, likert scales, interview protocols, as well as binary method fragment reviews.

2.04 Method Evaluation

The developed method fragment pool, the method selection rationale, as well as the method assembly and calibration principles are being evaluated by means of ten semi-structured expert interviews. Following the interviewee selection (see Sub-section 2.02.2), all experts were asked to respond to questions covering:

- General fragment suitability for vitality assessment
- Applicability and ease of use of fragment assembly principles
- Suitability and applicability of the presented fragment selection rationale
- Suitability and applicability of method fragment calibration
- Ease of use of the approach for software product managers
- Obstacles and difficulties in utilizing the method for vitality assessment

Furthermore, all interviewees were presented with thee use-cases describing different selection scenarios for situational method design (see Appendix B). Next to the above questions, they were asked to assemble a situational method, matching the presented situational factors and describe positive and negative aspects of their experience with the design process.

2.05 Design principles

Following the information systems research framework introduced by Hevner, March, Park and Ram (2004), republished in the book “Design Research in Information Systems” (A. Hevner & Chatterjee, 2010), this thesis’ research method leans on two important paradigms of information science: behavioral science and design science. Following this framework, the research model, introduced within the following section, combines the existing knowledge base of SECO related research with the business needs and technological specifics of the research environment. Hereby, the goal of the research is to produce an artifact, in the form of an OSC health analysis method, which will add to the existing SECO knowledge base and also be applicable in the business environment of software product management.

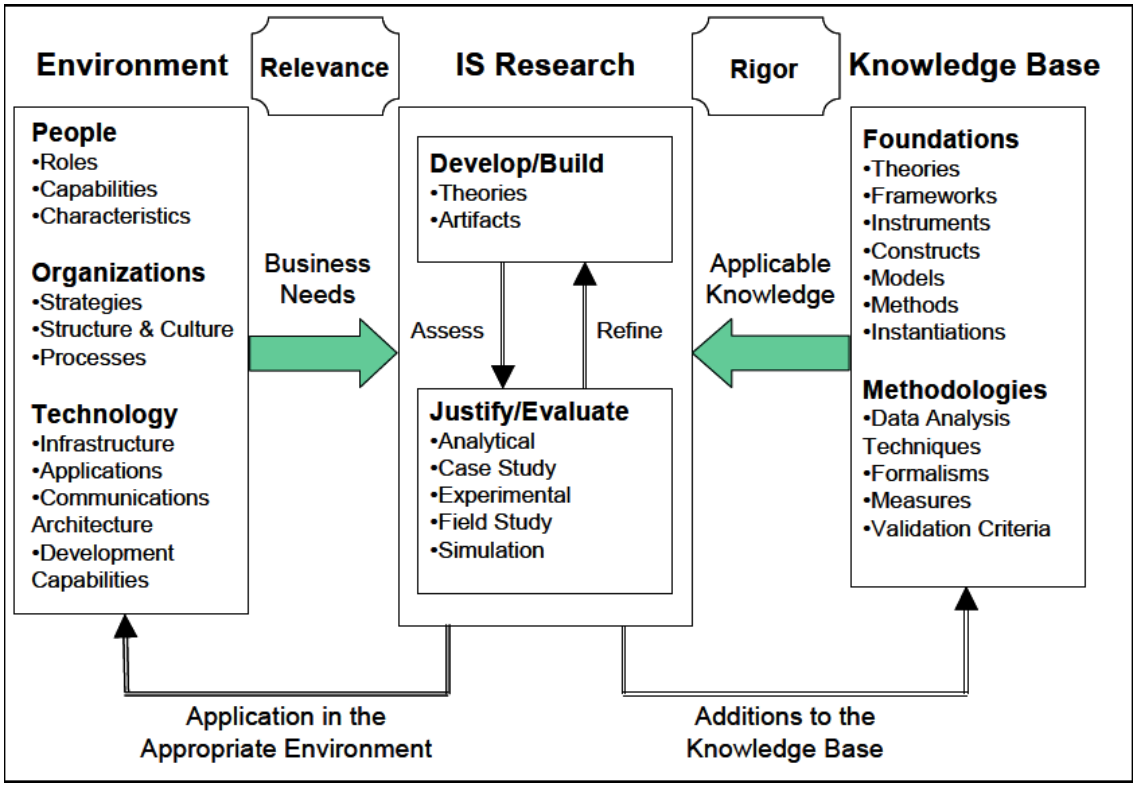

Figure 2 - Information Systems Research Framework by Hevner et al. (2004)

3. Theoretical Background

Based on a rigor scientific process, this section introduces the necessary literature background for this research. Placing the research question within the scientific body of knowledge, it becomes evident that it spans across three distinct information science domains: Open Source Communities, Software Product Management and Software Ecosystem analysis.

Within this document, OSC related topics are given a priority over the other domains, as their understanding is of the greatest importance for creating the thesis deliverables.

Following a general introduction into contemporary research on SECOs and an outline of core activities of software product managers, a systematic literature review, aimed at research approaches covering SECO or OSC health evaluation is conducted. The literature review serves as the theoretical foundation for the development of the OSC health analysis method and further contributes to the understanding of OSC vitality indicators, as well as open source participation models.

3.01 The Software Ecosystem Domain

In regard to the scientific body of knowledge on software ecosystems, the domain is increasingly growing in size and diversity. In general, software ecosystems can be considered a natural evolution of traditional software product lines (Bosch, 2009; Boucharas, Jansen, & Brinkkemper, 2009) and usually evolve from mature software products which had reached the limit in expertise or resources of in-house development. Although relatively novel (first appearances in 2005), many research contributions have been made to this domain.

Typical examples of such contributions focus on the analysis of selective ecosystem, such as the contributions of Ververs, Bommel and Jansen (2011) or Kabbedijk and Jansen (2011). Ververs' work is based on an empirical analysis of the developer participation in the Debian ecosystem. In order to understand the members' motivation, the author quantitatively distilled and analyzed influential factors for participation. Kabbedijk's and Jansen's research was focused on an analysis of the Ruby software ecosystem. By looking at Git commits of so called Ruby Gems, the authors created a visualization of developer interactions and characterized typical profiles of software developers within the ecosystem.

In contrast to case based analysis, other approaches lay their focus on formal methods or systematic analysis. Taking a reverse engineering approach, Lungu (2008) looks at individual project components and their relationships in order to create a meta perspective of a given ecosystem. Based on visualization, top-down exploration, architecture recovery and software evolution analysis the author proposes a method for systematically reverse engineering software ecosystems. Attempting to formalize ecosystem modeling, Yu and Deng (2011) rely on the *I modeling framework* for developing a better understand of software ecosystems. Hereby, the *I-models* focus on illustrating strategic dependencies between end-users, software vendors and third party developers. Furthermore, the authors draw comparisons between traditional software supply networks and ecosystem driven models. A different analytical approach is used by Dhungana, Groher, Schludermann, and Biffl (2010). The authors compare natural ecosystems to software ecosystems in order to present an agenda for further research. This is done by analyzing overlapping key

characteristics, present in both types of ecosystems. Also focusing on contributing to future research on software ecosystems Barbosa and Alves (2011) performed a systematic literature review of the domain. Mapping the ecosystem perspective, the study identifies 8 main areas in which the most relevant ones are open source software (OSS), ecosystem modeling, and business issues.

In contrast to exclusively focusing on formalization or ecosystem comparisons, Jansen, Finkelstein and Brinkkemper (2009) look into the perspective of software vendors transitioning to ecosystem oriented development. As a result of their analyses, the authors propose a research agenda on software ecosystems which includes, both, the technical and the business aspects of software engineering within contemporary software ecosystems. Similar to the attempts of the authors above, Bosch (2009), as one of the first in this domain, focused on software product line companies, transitioning, beyond their historic boundaries, to ecosystem driven models. For developing a better domain understanding, Bosch introduced a definition of software ecosystem, as well as the matching domain taxonomy.

3.02 Responsibilities and Activities in Software Product Management

Focusing on the strategic value of commercial software applications, the role of software product managers often involves activities, such as portfolio management, product road mapping, release planning and requirements management (Van De Weerd, Brinkkemper, Nieuwenhuis, Versendaal, & Bijlsma, 2006). Next to the internal development team, typically including development and support, SPMs must interact with a number of other relevant stakeholders. According to Brinkkemper, Ebert and Versendaal (2006), these can include internal research and innovation, partnering software companies within the software ecosystem, sales and marketing, as well as customers presently using the managed software product. Further responsibilities of SPMs include monitoring the capabilities of the software product among competing offerings and making strategic decisions in regard to future software capabilities (Berander, 2007).

Looking at contemporary SPM research, the work of Botzenhardt, Maedche and Wiesner (2011) contributes to developing a conceptual understanding of SPM activities by means of a domain ontology. The authors define domain-specific knowledge and list typical PSM activities associated with successful product management.

Reporting on the work of the International Software Product Management Association (ISPMA) towards establishing software product management as a discipline of its own in academia and within the software industry, Gorschek and Kittlaus (2011) present the current state of the curriculum and a certifiable body of knowledge on SPM.

In contrast to exclusively focusing on SPM Manteli, Van De Weerd and Brinkkemper (2010) studied the relationship and dependencies between software project management and SPM activities. Based on the results of a number of interviews and an online questionnaire, the authors answered a set business issues regarding the leadership roles of product and project managers within the organizational structure of software companies.

Focusing on a perceived gap between software product management activities and agile software development methods, Vlaanderen, Jansen, Brinkkemper and Jaspers (2011) propose and a new method

applying SCRUM principles to SPM activities. Based on a conducted case study, the authors illustrate how their method can enable product managers to cope with large requirements in agile development set-ups.

3.03 Systematic Review: Open Source Community and Ecosystem Health Research

In order to solidify the theoretical foundation of this thesis, a systematic literature review on the topic of OSC and SECO health was performed. Following the principles of designing an explicit, comprehensive and reproducible review, the below approach is adopted from the eight step guide towards conducting a systematic literature review of information systems research (Okoli & Schabram, 2010).

Hereby, the review is segmented into seven consecutive steps that systematically describe the taken approach from defining the review's purpose up to the analysis of the aggregated data.

3.03.1 Review purpose

The primary purpose of this review is to lay the scientific foundation for the consecutive steps of this thesis project. Furthermore, the results are intended to contribute towards designing interview protocols for qualitative data aggregations, as well as evaluating known vitality indicators for OSC health performance. In addition, the review enables us to place the scientific contribution of this thesis within the discovered contemporary SECO/OSC literature and enables comparison of research results with related work.

3.03.2 Systematic review approach

Given the systematic nature of this review, the approach is structured in clearly predefined consecutive steps that are divided into four main activity categories: Planning, selection, extraction and execution. A visualization of the process can be found below in Figure 3.

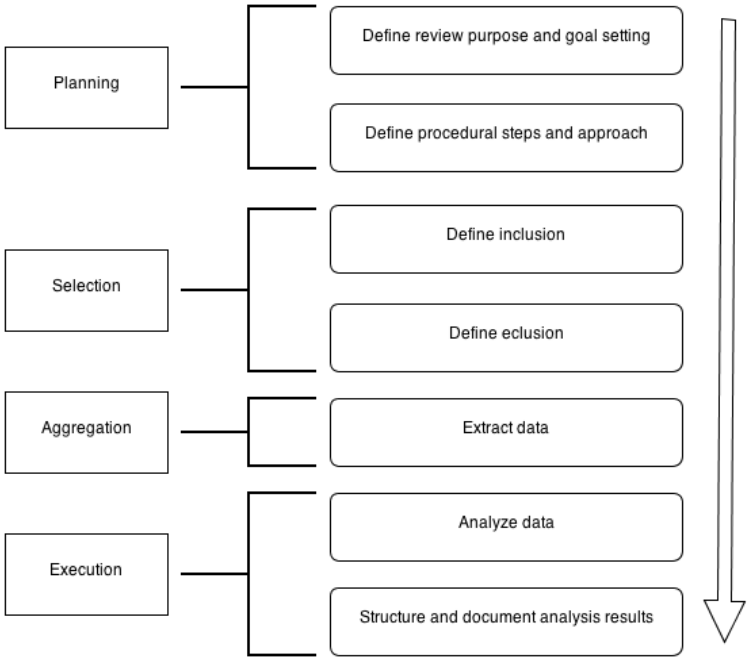

Figure 3 - Systematic Literature Review Approach

3.03.3 Inclusion criteria

The focus of the analysis lies on finding suitable scientific research that covers the domains of software ecosystems or open source communities in combination with an analytical approach towards OSC vitality or health related aspects. The intention is to present a holistic view on contemporary research studying i.e. ecosystem driven models, mapping out vitality related aspects of open source communities or focusing on defining or analyzing vitality aspects. Additionally to the general focus, all papers directly addressing one of the sub research questions are also included.

Figure 4 illustrates all search terms used for queries on the selected publication portals. Herby, all queries were executed with a combination of each term in the left column with each term in the right one. Due to this approach, 36 unique queries were performed.

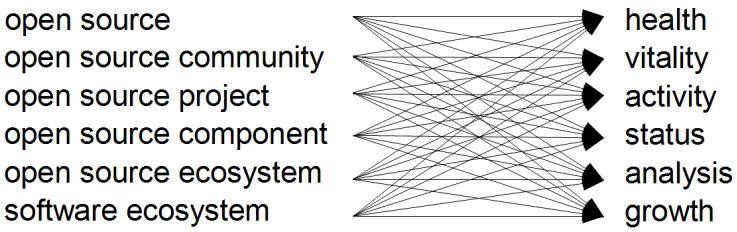

Figure 4 - Systematic Literature Review: Keyword combinations

A search with all keyword combinations has been conducted on, both, Google scholar and Microsoft Academic search. Together the two platforms cover a vast majority of all academic publishers and repositories. Both utilized search tools are capable of discovering free, as well as most subscription based scientific repositories.

3.03.4 Exclusion Criteria

The review is not intended to cover papers that exclusively focus on open source without ecosystem or health related perspectives. Furthermore, papers solely describing the SECO domain, research agendas and similar work are not targeted, as these lack relevance for answering the thesis' research questions. Furthermore, a limitation to the first five pages of result pagination is applied. Due to the use of popular search terms, such as 'health' or 'open source', publication search engines can, at times, produce more than 25'000 results. Due to such manually not procelssible numbers, this exclusion criterion is designed with the assumption that the utilized ranking algorithms weigh papers, including a combination of multiple relevant search terms, greater than those only matching one keyword. Based on additional filtering features, search results on Microsoft Academic are limited to the domains of Computer Science and Economic & Business. Results on Google scholar exclude results for patents. Further restrictions apply to language. The search was performed exclusively in English, articles in other languages than English, German or French were discarded.

3.03.5 Data extraction

The extraction process was performed as follows (see Figure 5). First, the titles of all scientific research (within the first five pagination results of Microsoft Academic and Google Scholar) were read. Titles matching the inclusion criteria, or those in question to potentially match the criteria were selected for closer review. Given the popularity of the individual keywords, only a small percentage of the discovered research was deemed relevant, as a vast number of research papers belonged to the domain of medical

informatics or did not match the predefined criteria. Consequently, each abstract of the remaining selection of 121 titles was read.

Having gained a deeper understanding of the research content of each paper, another 69 candidates have been discarded. The remaining 52 papers were fully read and further 22 discarded based on a lack of focus on OSC understanding, OSC vitality relevance or a sole focus on software ecosystems, without relevance to open source. A table listing all 121 titles passing the initial selection round (including the respective acceptance stages) can be found within the appendix of this document.

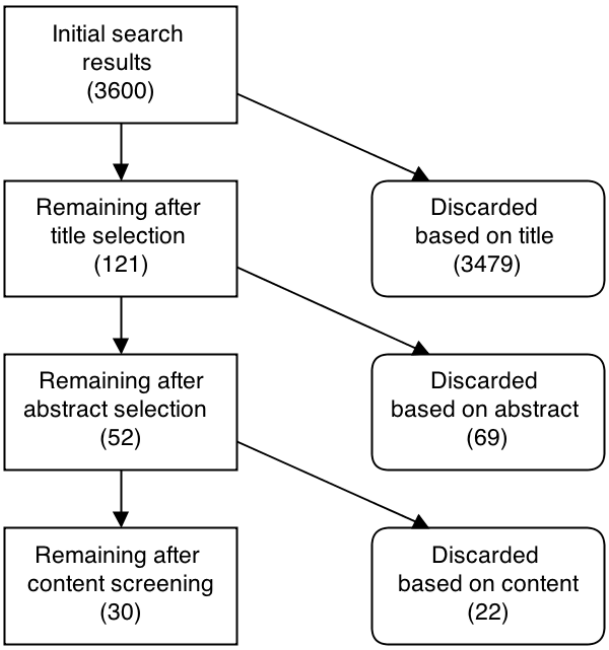

Figure 5 - Systematic Paper Selection Process

3.03.6 Analysis and results

The following section presents a categorized overview of the discovered state of OSC vitality research.

Community practices

Michlmayr (2005) researched the connection between software development practices of OS communities and community health. The author's hypothesis was that communities that practice more mature software practices, such as the utilization of version control, dedicated mailing lists, documentation guidelines for source code, systematic testing prior to a version release and software portability are more likely to be found in successful OS projects. After a comparison of 80 successful and unsuccessful OS projects with similar attributes on SourceForge, the author concluded that the incorporation of version control, effective mailing list communication and various testing strategies are the most likely indicators of a successful OS community.

After studying best practices and scientific literature on patch contribution processes in OS communities, Sethanandha, Massey and Jones (2010) proposed a standardized approach, as a measure for quality assurance of OSCs. Further aims of the proposed process were to enable better learning opportunities and knowledge transfers within OSCs and to improve the recruiting of new developers into the OS communities.

Applying a set of data mining techniques to a large number SourceForge projects, Raja and Tretter (2006) evaluated the suitability and implications of Logistic regression, decision tree and neural network analysis on predicting OS project success. Their findings were that due to an improvement in OS management processes and a rise in polarity, projects launched after 2003 were more likely to succeed than their older counterparts. Furthermore, number of downloads and reported bugs positively correlate with project success. This is explained with the fact that reported bugs stem from an active community and therefore indicate OSC popularity.

Specifically focusing on the activities of the core group of OSCs, Torres, Toral and Barrero (2011) utilize social network analysis techniques to evaluate the role this particular group plays within OS projects. The authors distinguish between core members, active developers and peripheral developers and identify core group activities that extend beyond code contribution. The authors particularly highlight the core group's importance for OSC self-regeneration and project evolution.

Looking into community practices over time, L. Yum Ramaswamy, Lenin and Narasimhan (2009) performed a time series analysis on a mailing list archive, two bug archives and one revision history repository. The authors discovered three types of frequently occurring patterns: seasonally dependent, cyclic but seasonally independent and acyclic activities.

Community and project popularity Assessment

The work of Wahyudin, Mustofa, Schatten, Biffl and Tjoa (2007) directly addresses OSS community health. Based on an evaluation of scientific literature on OS communities and software project monitoring, the authors constructed a software community interaction model, focusing on the three core quality related perspectives of OSCs: The developer community, the user community and the software product. Consequently, the authors constructed two core health indicators that aggregated previously introduced performance variables: Developer contribution and bug service delay. Validation was performed by means of data analysis on four representative Apache foundation projects. The authors conclude that their method touches upon important health indicators, however that many yet undiscovered factors also need to be taken into consideration. They further state that the relative weight of individual health variables varies between different OS projects and that each variable's relative importance would benefit from adjustment based on advice by project leaders of respective OS communities.

It is noteworthy that the work of Wahydin et al. (2007) can be seen as an extension to a previously published article with the title "Introducing 'Health' perspective in open source web-engineering software projects, based on project data analysis" (Wahyudin, Schatten, Mustofa, Biffl, & Tjoa, 2006).

Weiss (2005) takes a different approach to assessing OSC popularity and proposes to look at web search engine results. Based on the assumption that a successful OS application will enjoy broad distribution, he proposes four measures of assessing a communities' popularity, based on the number of matching search results. The findings indicate that only two measures demonstrate relevant and sensible properties, however the author advises against strong conclusions, due to uncertainties regarding the *black box* approach of search engine ranking algorithms.

Inspired by the concept of the tragedy of the commons, Schweik and Amherst (2007) conducted research on categorizing criteria for defining success or abandonment of *free libre and open source software* (FLOSS). Following a literature analysis and interviews with representative open source developers, the authors

developed and validated a six-stage classification system of FLOSS commons. The classification was applied to all OSCs hosted on SourceForge and was proclaimed to have an error rate of 11.5%.

The work of Izquerdo-Cortazar, Gonzalez-Barahona, Robles, Deprez and Auvray (2010) presents some of the more recent OSC vitality related publications. The authors statistically analyze OSC evolvability and robustness of 1400 FLOSS communities. Hereby, the utilized approach divided the main research question into two sub-questions dealing with: Size and regeneration, as well as interactivity and workload adequacy. The authors' concluded that their work demonstrated how OSC quality can be measured from an industrial perspective; however they also raise concerns regarding a potential lack of suitable information for the analysis.

Opposite to the efforts of Izquerdo-Cortazar et al. (2010), Samoladas, Angelis and Stamelos (2010) evaluate probabilities for FLOSS survival. By mining an open source database, funded by the European Union (FLOSSMetrics), the authors review time series data and attempt to predict the continuation of OS projects. This was done based on inputs, such as a project's application domain or the number of its committers. One of the highlights among the authors' findings is the observation that especially those projects that have existed for more than five years are very difficult to be abandoned. Furthermore, the authors' pessimistic calculations estimate an average survival probability of 40% for projects younger than five years. Additional findings indicate that for every new committer added to a project, its survival probability is increased by 15.8%.

Among all reviewed research papers, one of the most comprehensive approaches to OSC vitality was published in 2009 in the Elsevier Journal (Subramaniam, Sen, & Nelson, 2009). The paper addressed OSC project success by means of a longitudinal data analysis of OS projects on SourceForge over a time period of 5 years. Based on a literature analysis, the authors constructed a model of OSS success measures that was divided in developer interest in the project, user interest in the project and project activity. In addition, the model evaluated interrelations between success factors and introduced a segmentation into time dependent and time independent variables, such as e.g. developer interest or operating system language. The authors main findings are that project activity and developer interest were always positively associated and that user interest always affects all OSS performance measures of the consecutive period. Another conclusion was that persistent user-interest, on a long term, has a negative effect on project activity. Further discoveries were the growing acceptance for widows based open source projects and the negative effect of restrictive licenses on project activity levels.

Published as early as in 2003, the research of Crowston, Annabi and Howison (2003) on OS project success is one of the oldest scientific contributions within this review. Similar to other reported approaches, the authors attempt to measure and analyze OS success by means of OS developer interviews and a systematic literature review. The discovered success measures were project output, the development process and participation benefits for developers. Interestingly, despite its age, many of the findings match the results and opinions of recent OSC publications.

Another contribution to OSC health analysis is the work of German and Mockus (2003). The two authors worked towards automatically measuring the performance and activity of OS projects by designing a tool that would automatically retrieve, summarize and validate data from CSV logs, mailing lists, change logs

and bug tracking systems. The research result was an early prototype of a tool labeled *SoftChange* and the verdict that automated OS measurement tools were yet in their infancy.

Looking into OS developers' motivation to participate in OSCs, Wu, Gerlach and Young (2007) developed a theoretical work motivation model and validated it by means of a field survey that was answered by more than 140 OSS developers. The results of their research show that OS developers are motivated by a combination of altruistic and economic incentives, related to enhancing their personal level of expertise. Furthermore, the authors discovered a great diversity in overall satisfaction and career advancement opportunities among the questionnaire respondents. This led them to the conclusion that not all of the OSS developers benefit equally from their community contributions.

Similar to a number of the above scientific contributions, Crowston, Anabi, Howiston and Massango (2003) performed research on the definition of project success of OS software. The authors mined and analyzed projects on SourceForge and identified the following success factors: Number of members of the extended development community, the project activity, the avg. bug fixing time and the total number of downloads.

Commercial community sponsorship and licensing

By comparing corporately sponsored and individually founded OSS projects, West and O'Mahony (2008) identified three dimensions that affected member participation: Production organization, community governance and intellectual property management or licensing. Furthermore, the author introduced a participation architecture for corporate founded communities and identified a tendency of such communities towards more transparency and less openness. Some of the other findings included the insight that corporate incentives for community foundation were often based on marketing reasons and not code innovation. Furthermore, the authors drew a direct connection between a firm's interest to attract direct code contributions and its approach towards offering accessibility to the community.

Looking at similar factors as West and O'Mahony, Stewart, Ammeter and Maruping (2005) researched the effects of corporate sponsorship and restrictiveness of the licensing model on OSC popularity. After analyzing data from Freshmeat.net and representative OS project sites, the authors conclude that organizational sponsorship has a positive impact on project popularity, as it often associated with greater software quality. However, restrictive licensing models negatively affect OSC popularity due to a perceived reduction in the applicability of the OSC.

Software quality assessment

Performing an adoption of the Weibull distribution analysis on a time series of bug tracking data of OSC projects, Zhou and Davis (2005) drew parallels between closed and open source development practices. The authors conclude, that OS projects often exhibit similar reliability growth patterns as closed source examples and that mature projects tend to reach a similar, low stability point in bug reporting. Working towards a general reliability model for OSC projects, the authors find that existing models can and should be adapted to OSC evaluation. Another noteworthy discovery was the fact that neither page views nor download rates on OS portals highly correlate with a project's software quality.

Taking a component driven perspective, Maki-Asialaand Matinlassi (2006) studied OSC integration into software products and particularly software product lines. By conducting a set of interviews with Finnish software development firms, practicing OSC integration, the authors researched OSC evaluation techniques, applicable to commercial application development. The interviews discovered common

practices of static component testing and an aversion to formal test documentation processes. Several companies reported cases of swift OSC community responses in case of discovered bugs. In regard to testing preferences, a majority of the interviewees stated a preference for executable test and test benches. Some of the major benefits of OSC integration were reported to be white box testing and vendor independence. The greatest downsides were the multitude of licenses and considerations regarding uncertainty of code quality.

Similar to Matinlassi's work Hauge, Osterlie, Sorensen and Gerea (2009) focus on developer's rationale for OSC integration in commercial software products. The authors conducted 16 interviews among representative Norwegian software companies and conclude that presently, in practice, normative selection methods or general evaluation schemas have not been adopted within the industry. Furthermore, project specific constraints were reported to almost always outweigh general selection criteria for OSCs. The authors state that the 'first-fit' principle was often seen as the driving principle for OS evaluation.

The work of Immonen and Palviainen (2007) focuses on evaluating the trustworthiness of OS components that are intended to be integrated in larger software products. Aiming at the activities of software integrators, the authors developed a method and the matching Eclipse based tool set for evaluating OSCs in question. The developed method analyzes OSCs directly, as well as indirectly by looking at technical evaluation and component testing. Additionally, it takes qualitative indicators, such as developer or component reputation in consideration. The developed tool (RAP) helps developers with the analysis of OSC reliability on architectural level and supports integration testing.

Although primarily aimed at assessing OSS quality, research by Parizi, Azim and Ghani (2010) also provides valuable insights of OSC vitality analysis, as it approaches quality from a statistical time series perspective. To be more specific, the authors combine the domain of time series analysis with the one of software quality assurance and thereby use statistical data to predict and forecast OSS quality. Although this introduced hybrid method is promising, the authors' work lacks empirical validation, as only a theoretical model was introduced within the publication.

Another contribution to OSC software quality assessment is the SQO-OSS quality model (Samoladas, Gousios, Spinellis, & Stamelos, 2008). The model specifically focuses on automatically measurable metrics and aggregates a variety of variables into core evaluation criteria. The model was designed with the intention to be integrated into a decision support system and facilitates a profile based algorithm for its calculations. Despite initial positive results, a validation of the method was not part of the publication and remained an open task for future research.

4. Understanding Open Source Communities

Prior to introducing the OSC health method, this chapter focuses on creating an understanding of OSC participants and their activities and roles within OS communities. Hereby, it is not the authors' intention to present all aspects of open source in detail, but to specifically focus on those relevant for OSC vitality. Following the introduction of an OS community interaction model, we present a list of identified vitality indicators for OSC health and elaborate on risks, benefits and implications of commercial patronage for OSC vitality.

4.01 Open Source Participants

The following sub-section introduces an overview of typical open source participants and presents their typical activities and roles within open source communities.

4.01.1 Development Core Team

The core team plays a vital role for the health of any OSC and in most cases consists of one or a small number of developers who steer and orchestrate a communities' activities and strategy. Traditionally, core team members have commit privileges to the software versioning repository and play a pivotal role in architectural decisions. Next to their contributions to the development process, they are typically very active on community discussions, play an important role in attracting and integrating new developers and invest a considerable amount of time into their community activities. Both, a number of articles within the reviewed scientific literature, as well as interviewed domain experts stated the number of core team members and the core team stability as indicator for community health (see Table 3).

4.01.2 Active Contributors

This role is categorized by developers who make regular contributions to the OSC code base, however remain less engaged with the community than core team members. Typically, these developers participate in discussions regarding the project's development and its strategic goals; however their influence on community decisions weighs less than that of core members. Furthermore, they often do not have direct commit rights to the main code repository and are less involved in community communication and the recruitment of new developers.

4.01.3 Casual/Peripheral Contributors

In contrast to the first two developer categories, peripheral contributors are not frequent participants in community activities but only sporadically contribute code, respond to forum posts or the mailing list. Such casual contributions are often triggered by bugs or by missing features that have been discovered and fixed by the developer in question. Although valuable, these contributions do not follow a regular pattern and a community cannot rely on the dedication of such developers.

4.01.4 Community members (Non-Development)

Next to the frequently mentioned developer roles, open source communities also consist of individuals who contribute to the project's success by other means than submitting source code. Such individuals

participate in discussions on mailing lists, IRC or online forums and provide support and advice to other community members. Further activities can include contributions to the project documentation, formulating feature requests or promoting the OSC outside of its community scope. Arguably, the most important contribution of non-developing OSC member is software testing. Active communities have proven to be thorough testers of new OSC software releases and generate the majority of a project's bug reports, thus greatly contributing to OSC software quality (Raja & Tretter, 2006).

4.01.5 Commercial Patronage / Sponsorship

A relatively novel part of community participation is commercial patronage or sponsorship. Alike to the presented examples in Section 5, an increasing number of open source communities have strong ties to commercial software vendors. This growing influence becomes most evident when looking at documents, such as the annual Kernel Development report ("The Linux Foundation Releases Annual Linux Development Report," 2012) published by the Linux Foundation. Listing the top ten contributors to the Linux Kernel, the list entails a number of enterprises, such as IBM, Oracle or Nokia.

Next to official reports published by OSCs, the increasing role of commercial interest in open source is portrayed by a number of contemporary scientific articles, highlighting its effects on OSC popularity, transparency and accessibility (Stewart et al., 2005; West & O'mahony, 2008). Hereby, types of contribution can vary from small financial or source code donations up to top-down community foundation and considerable degrees of involvement.

4.02 The Community Interaction Model

By adopting aspects of the "casual model of an OSS project" by Wahyudin et al. (2007), this subsection introduces an OS Community Interaction Model that focuses on four identified core aspects, relevant for OSC vitality: OSC Software Quality, Commercial Patronage and Involvement, User Community Vitality and Developer Community Vitality.

The introduced aspects and forces of influence within the model (see Figure 6) are distilled from the systematic literature review, as well as the performed expert interviews and include the most frequently mentioned OSC interactions and their respective influences on each other.

The model starts with OSC Quality that is comprised of release frequency, bug response time and code quality. The intention of this separate view is to create an understanding of the mutually dependent forces within an open source community, which determine software quality and ultimately shape an OSC's relevance from the integration standpoint of a software product manager.

Code quality and release frequency both have a positive impact on user community vitality as they affect user satisfaction and thus promote frequent use of the OSC. In return, OSC use or OSC popularity within the community are positively associated with the amount and quality of the generated user feedback, which consequently affects the developer motivation and the vitality of the developer community.

Next to user feedback, developer motivation is dependent on project accessibility (ease to interact with OSC developers and to contribute code) and additional support, such as financial or legal aid, provided by a commercial party. Motivated developers generate code contributions, which in return determinate OSC software quality.

With increased regularity, commercial parties are an important participant in popular OSCs (Stewart et al., 2005) and are thus depicted as a separate aspect of the interaction model. Hereby, their participation can be described by their degree of involvement in OSC activities. A strong presence of commercial patronage within OSCs has been found to both improve community transparency and negatively affect community accessibility. Other types of involvement, such as commercial and legal support or code donations have been found to positively contribute to OSC code quality or developer motivation (West & O'mahony, 2008).

Looking at the overall model, the complexity of the inter-dependencies between the different players and their respective motivations become evident. A community's vitality must thus be always evaluated in light of the impact a single change has on the general picture. Furthermore, it becomes evident that individual aspects within the model can be seen as a concatenated dependency path. This implies that changes to one particular motivator do not just influence the immediately linked association but can cause a chain of events, affecting the vitality of the entire OSC.

Additionally, the model shows that a number of stimuli can affect more than one vitality aspect at the same time. For instance the impact of community involvement positively correlates with both, code quality and project transparency. Yet, it has a negative impact on project accessibility. Thus, it is important that all dependencies of a given OSC force are reviewed in order to assess its implications for the overall community vitality.

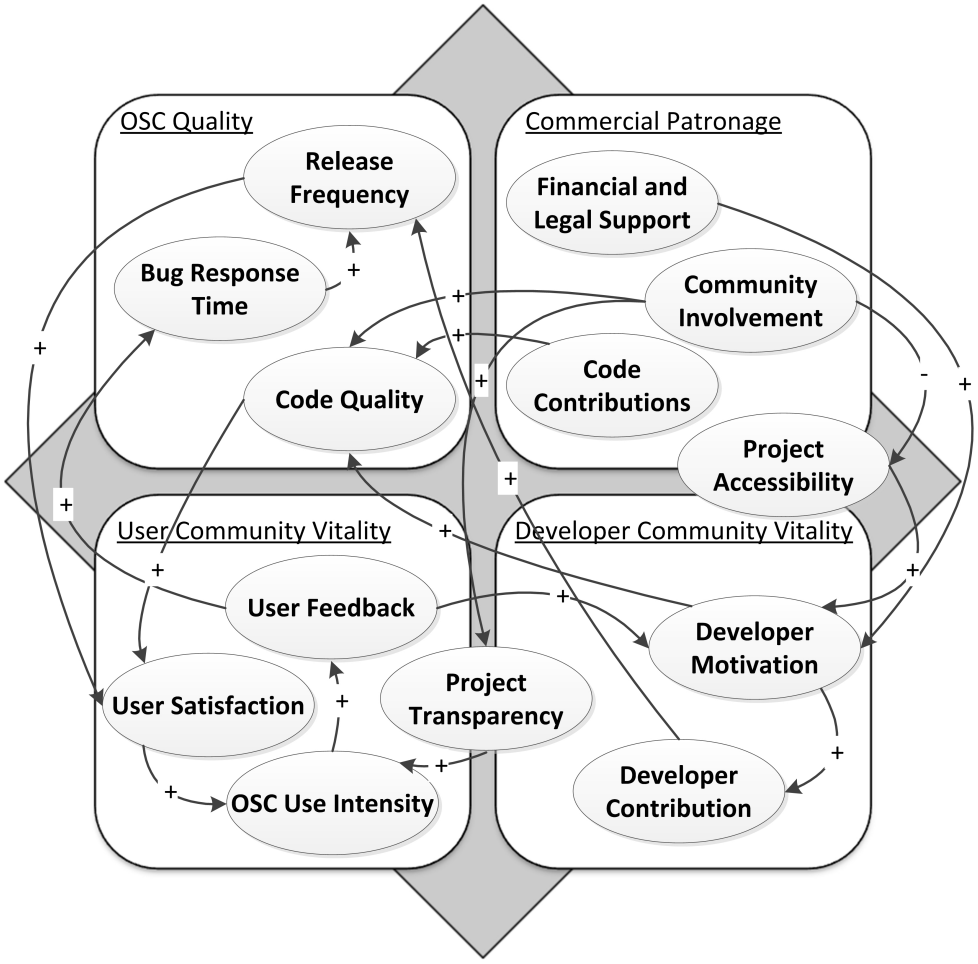

Figure 6 - Open Source Community Interaction Model

4.03 Community Health and Vitality Indicators

The following section introduces a collection of theoretical and empirical OSC vitality indicators, originating from the performed systematic literature review and the previously introduced expert interviews, conducted at IBM.

Table 3 depicts the indicator names, descriptions and their particular origin(s). An anonymized list of IBM interviewees and their respective fields of expertise can be found within the appendix of this document. The indicators can be seen as a high level conceptualization of a number of viewpoints or aspects of OS communities. It is important to highlight that all quantifiable attributes should be considered as relative values, dependent on respective OSC realities. For instance, online popularity or the total number of downloads can greatly vary between different types of communities. Despite low absolute indicators, a niche open source project with a small core team or a low number of total downloads can still be successful and healthy. This view is supported by recent OSC related publications that argue that the relative weight of vitality variables can vary between OS projects and that each variable should therefore be adjusted to the characteristic of its community (Wahyudin et al., 2007).

Next to considerations for absolute or relative variables, another controversial OSC vitality indicator is the type of OSC license(s) a project is using to publish its source code. The controversy stems from the fact that it is unlikely that a software project manager would consider the utilization of licenses that are not suitable for commercial software applications, yet all types of OSC licenses can have different implications for community motivation. Due to their importance for OSC vitality, this section introduces OSC licenses as a valid vitality indicator; however the following OSC health analysis method is based on the assumption that software product managers will only utilize OSCs that are published under commercially acceptable terms, such as the MIT license.

Finally, although covered by some of the reviewed literature approaches, no consideration is given to indicators, such as software category, the programming language or current trends in software development. Despite discovered correlations with community vitality (Katherine J Stewart, Ammeter, & Maruping, 2006), each case had to be adjusted to the local realities of the respective OSCs, contemporary trends and other non-generic factors. The authors find such adjustments to be difficult to reproduce and hence exclude them from the list of suitable vitality indicators.

ID	Vitality Indicator	Description	Origin
V1	Activity Patterns	• Yu et al. (2009) differentiate between: ○ Cyclic and seasonally dependent activities ○ Cyclic but seasonally independent activities ○ Acyclic activities	(Yu et al., 2009)

V2	Avg. duration before a bug fix	• Research with software companies integrating OSCs showed that the response rate for bug fixes can be seen as an indicator for community activity • Two interviewed experts reported to use this as an indicator for the vitality of OSCs they had used in the past	(Maki-Asiala & Matinlassi, 2006) Interviewees: 2 and 4
V3	Community age	• The longer a community exists, the less likely it is to be abandoned • Communities older than 5 years are considered mature and often exhibit similar software development processes as those practiced in commercial software firms	(Samoladas et al., 2010) Interviewee: 5
V4	(Core) developer reputation	• Technical and leadership reputation of (core) developers of OSC projects • Developer contributions within their domain of practice	(Immonen & Palviainen, 2007) Interviewee: 2
V5	Corporate Patronage	• Results of the study of Stewart et al. (2005) suggest that sponsored OSC projects are more likely to become popular than non-sponsored ones • Research by West and O'mahony illustrates that patronage affects transparency and accessibility of OSCs. Sponsored projects are often more transparent, however are less accessible than button up communities	(West & O'mahony, 2008) (Stewart et al., 2005)
V6	Developer Interest and motivation	• Developer interest can be defined by: o Altruistic reasons to contribute to a project o An opportunity to improve skills in a technical domain o Financial gains from new or improved skills in a high demand domain • Project activity and developer interest are positively associated • OSC user interest is positively associated with developer interest	(Subramaniam et al., 2009) (Wu et al., 2007)
V7	Frequency of bug reports	• Statistical evidence found by Raja and Trotter (2006) suggests that frequent bug reporting indicates an active community	(Raja & Tretter, 2006)
V8	License	• Restrictive licenses, such as GPL negatively correlate with	(K. Stewart et

	restrictiveness	activity levels of OSS projects • Stewart et al. observed marginally significant effects of license restrictiveness on the number of OSC subscribers	al., 2005) (Subramaniam et al., 2009) Interviewees: 1, 2, 3, 4 and 5
V9	Maturity of Software development processes	• Utilization of publically accessible version control • Structured patch contribution processes • Dedicated Mailing Lists or equivalent • Systematic testing • Formal release roadmap • Formal acceptance process for new developers to gain privileges of the core team	(Michlmayr, 2005) (Maki-Asiala & Matinlassi, 2006) Interviewees: 2 and 5
V10	Number of committers	• Absolute and relative number of code committers • Samoladas et al. estimate an increase in probability for project survival of 15.8 % for every new committer	(Samoladas et al., 2010) (Crowston et al., 2003) Interviewees: 1, 2 and 3
V11	Number of OSC downloads	• Total and relative number of downloads • Total and relative numbers of version management clones (e.g. via Git) • Controversial Indicator: *Opinions on this vary in literature. Research by Maki et al. indicated no correlation between OSC success and the total number of downloads*	(Raja & Tretter, 2006) (Maki-Asiala & Matinlassi, 2006)
V12	Online popularity	• Trending Topics on search engines (over time) • Frequency of cross references in technical blogs or by other software projects • Reputation within expert communities	(Weiss, 2005) (Immonen & Palviainen, 2007) Interviewees: 2 and 3
V13	OSS code quality	• Code inspections allow to approach uncertainties regarding OSC code quality and indicate skill level of OSC submitters	(Maki-Asiala & Matinlassi, 2006)

		• Research by Samoladas et al. (2008) indicates a direct connection between community practices and generated source code • Interview results highlighted that code screening /review is a common measure for OSC evaluation • A decrease in perceicved code quality can indicate changes in OSC vitality e.g. due to core developers leaving the community	(Parizi & Ghani, 2010) (Samoladas et al., 2008) Interviewees: 2 and 4
V14	Project activity	• Theoretical and empirical Indicators that can be utilized to define project activity are: ○ Mailing list activity ○ Forum activity (e.g. active topics or posts) ○ Avg. time to respond to feedback ○ IRC ○ Git discussions • Bug reporting	(Subramaniam et al., 2009) Interviewees: 1, 2, 3, 4 and 5
V15	Quality of OSC documentation	• Quality of functional OSC description • Quality of API documentation • Quality and usability of code examples	Interviewees: 2 and 4
V16	Release frequency	• Frequency of releases of minor updates • Frequency of major updates including added functionality	Interviewees: 1, 3 and 5
V17	Size of core development team	• The Core team is vital for community health. It drives and coordinates community activities. • Torres et al. differentiate between: ○ Core members ○ Active developers ○ Peripheral developers	(Torres et al., 2011) Interviewee: 3
V18	User interest in the project	• On short term, user interest positively influences developer interest and project activity • Long term influences have contradictory results. Data gathered by Subramaniam et al. (2009) indicated a negative long term influence on project activity while Stewart et al. (2006) report a positive correlation	(Subramaniam et al., 2009)

Table 3 – Open Source Community Vitality Idicators

4.04 Corporate Patronage in Open Source

One of the most noteworthy observations resulting from the operationalized data was the growing variety of different types of OS communities. As an example, next to common attributes, such as community size, many OSCs can be categorized by their types of origin, degree of corporate patronage or number of commercially affiliated developers.

The classical and best known OSC growth structure is based on the bottom-up principle, meaning that a community was founded by a group of developers or enthusiasts around a chosen topic and then grew by means of collaborative efforts and new members joining after having discovered their interest in the project. However, a number of large contemporary OSCs show top-down or hybrid structures. These are often characterized by one or many commercial stakeholders founding or significantly contributing to open source communities, thus becoming an important and influential partner within their collaborative structure.

Alternatively, commercial interest can be expressed by means of patronage or sponsorship. If a commercial software developer deems an OSC relevant to its business model, strategy or software development efforts, the organization can sponsor the community in question. In the reviewed case of the Apache foundation this was realized by means of financial contributions to the project, legal support and facilitation of face to face meetings. Such community support can have significant impact on community activity and is less intrusive than the top-down foundation approach.

Finally, organizational interest in OSCs can be measured by the number of commercially affiliated developers contributing to a community. Although developers participate and contribute code on an individual basis, many do this within their regular working hours using their business e-mail addresses, thus either having an implicit permission to work on the OSC by their respective employers or even being encouraged to do so. In the previously introduced case of the Linux Kernel, the Linux Foundation records the commercial affiliation of the contributing developers in the following way ("The Linux Foundation Releases Annual Linux Development Report," 2012):

"For each developer, corporate affiliation was obtained through one or more of: (1) the use of company email addresses, (2) sponsorship information included in the code they submit, or (3) simply asking the developers directly."

As mentioned in sub-section 4.02 and Table 3, recent scientific studies have evaluated the impact of corporate patronage on community vitality (K. Stewart et al., 2005; West & O'mahony, 2008). Next to the effects on intra-OSC interactions, illustrated in Figure 6, varying degrees of patronage have been identified to affect the following OSC attributes:

OSC attribute	Description
OSC transparency	The degree to which the internal decision processes and the development activities of a community are visible to the public
OSC Inclusiveness	The ease or complexity for an external developer to participate and contribute to OSC development activities
OSC Code Quality	Following the high level definition of the Consortium for IT Software Quality (CISQ) from an integration point of view, high quality OSCs are defined by their: • Reliability • Efficiency • Maintainability • Security • Conciseness
OSC Reputation and Popularity	OSC popularity within expert communities or among developers is positively affected, if the project is supported by a reputable or well acknowledged software developer (see Section **Error! Reference source not found.**)
Software Development Practices	Degree of formalization and structure of practiced software development processes within an OSC. This can be defined by practices, such as: • Defined release schedule • Systematic unit or integration testing • Formal requirements management and prioritization • A defined software development process e.g. SCRUM
Community Governance	The governing structure of communities and the degree of its formalization
OSC License Choice	OSC Licensing heavily influences the applicability of OSCs for software development firms. Commercial patronage is typically associates with less restrictive licenses, such as the MIT License (MIT)

Table 4 – OSC Attributes Affected by Commercial Patronage

5. Industry Insights in Open Source Practices

Following the introduction of a theoretical framework, OSC vitality indicators, as well as an open source interaction model, this chapter focuses on industry insights and practices at the facilitating case company. IBM has historic ties to, open standard development and implementation, as well as open source contribution and participation. Thus, the organization possesses great OSS expertise in general, as well as specifically in the domain of commercial open source utilization.

As early as 1999, the company formulated three key strategic goals for open source, which are still up-to date as of the writing of this document (Capek, Frank, Gerdt, & Shields, 2005):

- *"To support rapid adoption of open standards by facilitating easy access to high quality open-source implementations of open standards in order to speed industry adoption. A primary goal is to encourage open-source implementation of open standards and thus use open source as a way to support our business and strategic goals."*
- *"To use open source as a business tool by keeping the platform open and taking advantage of new business opportunities. By creating more opportunities, we encourage choice and flexibility in responding to customers' needs in typically heterogeneous environments."*
- *"To enhance IBM mind share, creating a preference for IBM brands by associating them with successful OSS projects and building relationships with a broad spectrum of developers. We contribute to key OSS projects that are functionally connected with some of our key products. The joint participation of commercial developers and independent OSS developers creates a synergy that enhances the open-computing ''ecosystem."*

5.01 Current OSC related practices

The following sub-section introduces IBM's current practices in managing its open source engagements. Hereby, IBM's approach towards internally utilizing open source components is documented in more detail and conclusions for the design of the OSC health analysis method are drawn.

5.01.1 The Open Source Core Team and its Activities

Known as the Open Source Core Team (OSCT), IBM has an internal cross-divisional group of managers, engineers, consultants, and attorneys responsible for managing IBM's engagement in open source software activities. The group's primary purpose is to provide advice and guidance on open-source matters and establish policies and best case practices for OSS utilization within IBM. In order to ensure a wide adoption and appropriate management of OS within the firm's global reach, each business unit is represented within the OSCT. One of the OSCT's main tasks is the implementation and the design of an open source review and approval process (OSSC), as well as the Open Source Participation Guidelines (OSPG).

OSS Usage Scenarios

The open source approval process is designed for a range of IBM interactions with open source software and covers five general scenarios:

- Including open source software within a product or service of IBM
- Interacting with and contributing to open source communities
- Including open source in service offerings, such as Software as a Service
- Third party vendor distribution (OSS is not distributed or linked directly to IBM products)
- Internal open source use, unrelated to IBM services or products

It is noteworthy that in particular the first two of the above mentioned scenarios can be divided in a number of use cases that all have varying approval processes. For instance, representative examples are IBM products that use OSS, third party software that IBM resells or Beta programs of software products that contain OSS. In regard to OS interactions, foreseen scenarios include creating a new OS project, joining existing projects with the intention of contributing code, code contributions without explicit project membership or the publication of code under an open source license.

Open Source Participation guidelines

All employees, interacting with any form of open source, must annually complete the open source participation guideline (OSPG) training. This can be done by attending an according information session or by reviewing the online training materials. Once completed, the employee is considered OSPG certified and is permitted to submit OS request.

The OSPG cover a broad range of OS topics and have a focus on legal considerations, such as distinguishing between risk free and dangerous licenses, implications of OSC distribution within IBM software and specific risks related to the use *"viral"* OS licenses, such GPL. A more detailed overview of the OSPG content can be found in Table 5.

General Topic	Content summary
Open Source Licenses	• Overview of the most common OS licenses and their implications from a commercial software perspective • Introduction to preferred OS licenses, such as the Eclipse Public License
Open Source Strategic goals	• Rapid adoption of open standards (see Section 5) • OS as a business tool, stimulating change in existing markets • Contribute to IBM's brand perception
Open Source Risks for IBM	• Risks originating from "viral" licenses that enforce open sourcing all software using such OSCs • Intellectual Property rights on commercial products using OS • Warranty Considerations for products relying on OSCs
Participation in Open Source Activities	• Distinction between private contributions and participation as a representative of IBM • Participation in OS communities competing with IBM software products • Clear separation between IBM source code and community work

Open Source Approval Process	• Internal Open Source approval process (see following section) • Approval Steps and Instances • Automated workflow system for open source management

Table 5 – IBM Open Source Participation Guidelines

General OSSC Approval Process

Before participating in OS activities or utilizing open source software, IBM employees must first undergo a specifically designed OS approval process. The process covers a wide variety of OS interactions and is designed to dynamically adjust to a number of potential risks to IBM. Regardless of the type of OS engagement, the process always follows a similar pattern and can be segmented into three consecutive phases:

1. Open Source Categorization
2. Determination of Approval Instances
3. Risk Assessment

Supported by an automated workflow management system, employees submit OS requests. Depending on the OS license and the component's potential impact on IBM, the system automatically assigns a review level for OSC validation. For each new submission, responsible technical and legal experts validate the OSC in question. Finally, the applicant receives notification of acceptance or rejection.

Review Levels

The number of required review steps and thus the complexity of the entire process are determined by the review level assigned by the workflow engine. Hereby, the value is mostly based on the intended OSC utilization and its software license. An overview of the levels and their implications for the review process can be found below.

Level 0	This review level is considered the lowest and is reserved for individual use of OS with Global Services engagements or for cases in which exclusively All Usage Approved (AUA) OSCs are utilized.
Level 1	Level 1 applies, if the utilization is well defined, fully transparent and involves components, or services that have already been reviewed. Furthermore, this level is a prerequisite for inclusion in IBM products.
Level 2	This level is reserved for OS reviews with a less well known scope than with Level one and OSCs with acceptable, known or unique licenses. Proposals on this level must be limited to the influence area of a single group within IBM's organizational matrix.
Level 3	OSCs not covered by the definition of lower levels are automatically placed on level 3.0 and above. Cases with a review level up to 3.1 are considered to affect the group level. Review levels above this value can escalate up to a mandatory Open Source Core Team (OSCT) meeting.

Table 6 – Open Source Approval Process: Review levels

Automated Workflow System for Open Source Management

All review processes for open source topics are supported by the automated workflow system for open source management (AWSOM). Next to facilitating the approval process and managing the review steps and activities, the system contains a database of all previously reviewed OSCs, including their respective release versions. This measure ensures that all process participants have a common overview of companywide OSC acceptance activities and prevents redundant OSC reviews. Another core feature of AWSOM is its capability to automatically determine the required review level for OSC inputs. Following the level selection, the system involves the responsible employees for the review process and concludes the process by returning the review results to the system manager of the OSC applicant.

5.01.2 Process for Open Source utilization in commercial software products or services

Following the introduction of the general OSC approval process, the diagram below (Figure 7) depicts its specific implementation in cases of open source utilization within commercial software products or services. As AWSOM plays a significant role in the review process, however only performs a supporting function, its activities are depicted separately in Figure 8.

The utilized notation originates from the research domain of method engineering and is based on the previously introduced process deliverables diagram (Van De Weerd & Brinkkemper, 2008). It focuses on both, the activity flow, as well on the according method deliverables.

A typical approval process is initiated by IBM employees who want to utilize open source within their development activities. After having completed the annually repeating certification of the OS participation guidelines, employees gain accesses to the AWSOM workflow tool and can submit OSCs for evaluation. If the component (in the requested version) had previously been reviewed within his business unit or by the OSCT, the process is considered complete and the developer can either proceed to using the component, or must accept its rejection.

If the component has not been reviewed yet, the developer must perform a so called pedigree review that consists of an online search for intellectual property claims related to the OSC, perform a code scan and review the components license(s). Along with the intended use of the component, the pedigree report is submitted into AWSOM that automatically assigns an according review level and generates a checklist for all involved reviewers.

Through AWSOM, technical and legal reviewers of the applicant's business unit receive review invitations and submit their evaluation into the workflow engine. Reviews with a level of 3 and below do not require cross group considerations and are returned to the applicant's system manager, who processes the review results and informs him about the decision. Review levels rated 3.5 or above can be considered highly sensitive, as they bear cross group implications. For such rare cases, on top of the group level reviews, a meeting of the open source core team must be held. The OSCT evaluation is then included into the workflow and passed on to the system manager to complete the workflow.

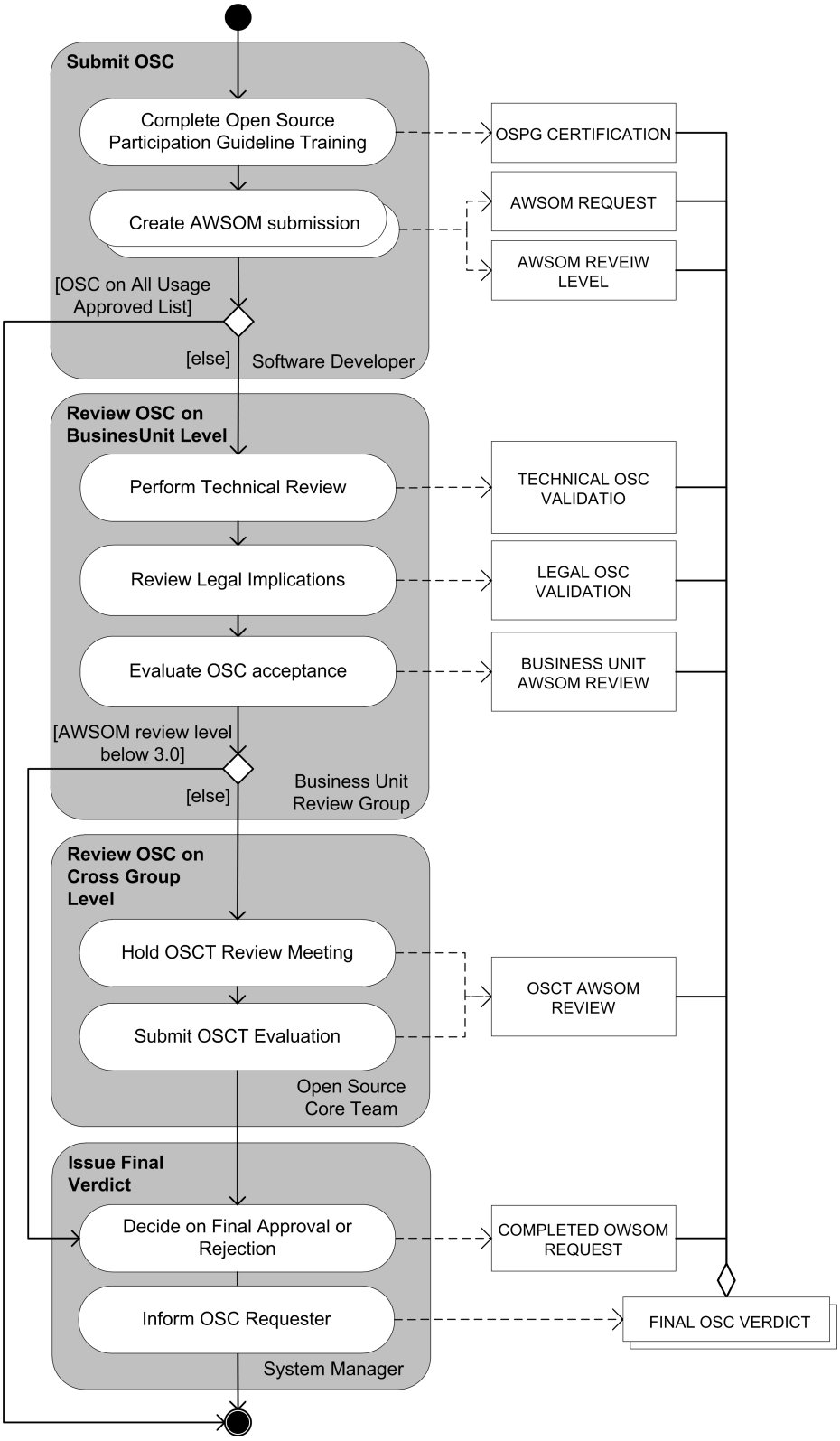

Figure 7 – Case Company Practices for Open Source Utilization in Commercial Products and Services

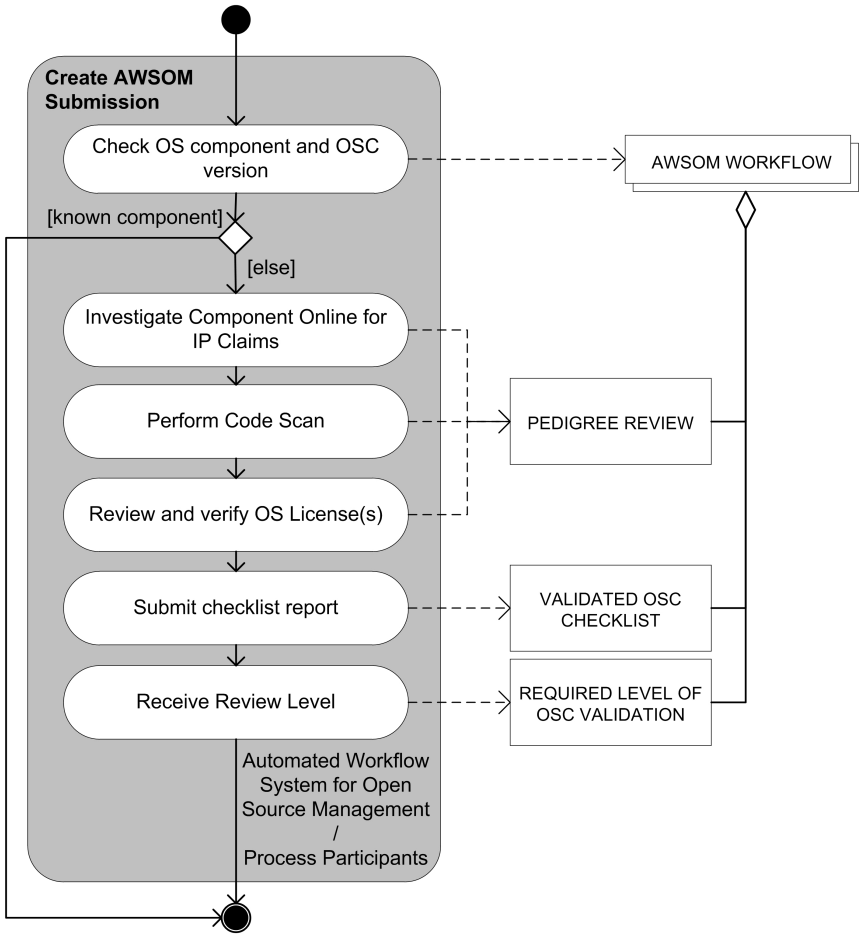

Figure 8 – AWSOM Workflow Engine, for Product Usage Requests

5.02 Learning points for the OSC Health Analysis Method

Looking at the thesis' research questions and the research aim, the current practices at IBM and the interview responses bear some initial learning points towards designing the OSC health analysis method.

The first and most obvious observation is that IBM's formal OSC evaluation process focuses almost exclusively on legal considerations, such as licensing or intellectual property. In its current state the process does not contain any activities reviewing software quality, the community structure it originates from or aspects of software maintainability. These topics are left up to individual developers or software architects.

This form of priority setting can be explained with a number of arguments, such as the vast financial liability risks, or the necessity of an international organization to control aspects of its software activities from a top-down perspective.

Despite a lack of applicable examples that could be used towards designing the OSCT health analysis method, the results of this chapter illustrated the growing importance of open source communities in commercial settings and presented examples of the impact of strategic OSC utilization. Furthermore, they provided insights into typical considerations that multinational enterprises must take prior to engaging in open source activities. This is particularly relevant when looking at OSC licenses that enforce publication of modified code and the great legal risks they have for products that are built upon them. Finally, it shows

that even within technology companies with a strong focus on software development, such as IBM, the licensing risks are still seen as the primary concern when using open source. This view is supported by the fact that the current OSCT processes do not cover any considerations for OSC software quality, maintainability, product lifecycle or support aspects.

6. The OSC Health Analysis Method

Built on the developed understanding of OSC vitality, this chapter introduces a systematic approach for software product managers that enables them to evaluate the vitality of candidate open source communities. Hereby, the foundation for the following method is based on five expert interviews (Section 2.02.2), the introduced scientific literature (Section 3), a study of OSC related practices at the case company (Section 5.01), the developed OS community interaction model (Section 4.02), as well as the distilled list of OSC vitality indicators (Section 4.03).

The authors find it important to highlight that the primary objective of the following health analysis method is not to present a universally applicable method that is suitable for any type of open source community or any kind of software company. Instead, the focus lies on developing a situational understanding of each case and then to select a number of method fragments that are best suited for the respective situation. Thus the aim of the health analysis method is to provide software product managers with a toolset that enables them to make the best assessment in a variety of scenarios.

Within this chapter we will introduce the concept and the features of a method fragment pool and method assembly (Harmsen, Brinkkemper, & Oei, 1994). These will be utilized to create situational methods for assessing OSC vitality. Furthermore, we present a selection rationale for choosing and assembling appropriate method fragments. Finally, the chapter contains a collection of varying fragments that can be used individually, or in combination to perform an OSC vitality assessment, as well as a sub-section for method calibration.

6.01 Method Fragment Pool

Derived from the domain of method engineering, this segment introduces the concept of situational method design, as well as that of a method fragment pool, containing base elements for method assembly. While the general principles of situational method engineering (Van De Weerd & Brinkkemper, 2008) fully apply, this section focuses on a customized adoption towards the specific needs of Software Product Managers and their responsibilities in regard to OSC dependency management.

Built upon the work of Harmsen, Brinkkemper and Oei (1994), Figure 9 depicts a process flow aimed at creating a situational method for OSC vitality assessment. Furthermore, the diagram includes an iterative cycle, focused on the customization of method fragments towards a community's unique characteristics.

The building blocks of a situational method are so called method fragments, which are stored within the method fragment pool. Each element within the pool focuses on assessing one specific OSC vitality aspect and can be either used by itself, or in combination with other fragments. Each fragment is linked to situational factors that make it best suited for assessing either a given community type or determining its applicability for a given Software Product (see Section 6.02).

Fragment selection begins with an assessment of OSC characteristics, necessary for narrowing down the total number of candidate method fragments for the given case. Combined with the requirements and the resources of the respective Software Product, both form the selection rational for choosing suitable

method fragments within the existing fragment pool. Sub section 6.02 contains a more detailed description of community and software product based fragment categorization.

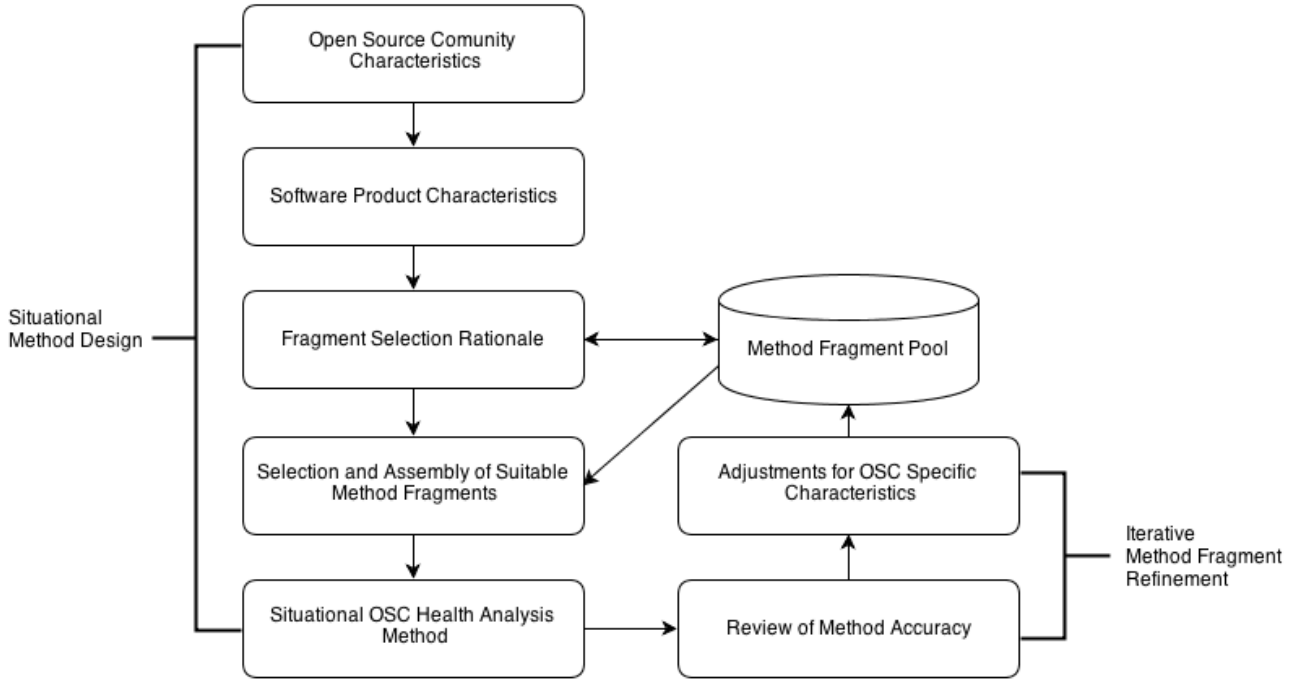

Figure 9 - Situational OSC Vitality Method Design Process

Following the selection process, the chosen fragments are being assembled into a situational method. Although a particular order is not a requirement, a logical structure or segmentation into fragment groups can contribute to the method's usability. Depending on the relative significance of community aspects, fragment importance can vary. Thus, looking at the overall quality of the software product, individual fragments can receive a greater weighted influence over less significant ones (see Section 6.04).

After method assembly, the individual fragments and the entire method can undergo an iterative process, aimed at improving the accuracy of the assessed OSC vitality. Hereby, historic or present community data can be used to review fragment and method accuracy. If a given fragment is found to generate inaccurate or contradictory vitality data it can be adjusted, its result weighed less or it can be fully removed from the situational method. All community specific adjustments can then be updated and re-inserted into the fragment pool. Should another software product, with varying project characteristics, require assessing this particular OSC, the new and more accurate fragments can be implemented.

6.02 Fragment Categorization and Selection Rationale

The following sub-section introduces a number of criteria that can be used to select situation specific method fragments for a given OS community or software development project. The main reason for this approach is the intention to create a practical heuristic that can be quickly utilized for method assembly. Furthermore, the approach does not enforce the selection of specific fragments, based on the presented rationale, but merely suggests using it as an indicator for assessing fragment suitability. The three introduced criteria have been selected based on their simplicity for the use by software product managers and their universal applicability for varying types of OSCs. Herby, the first two indicators stem from the

analyzed scientific literature and focus on specific community characteristics. The third indicator is based on the authors' estimate for required fragment resources and aims at a software product's development scope.

6.02.1 Community Size

The most evident indicator for fragment selection is community size. Hereby the size can be measured by the number of OSC developers, the number of core developers or alternatively the total number of registered community members. While this indicator needs to be adjusted for each community's nature or characteristics, it can help to quickly exclude fragments that are only applicable to large or very small OSC types. Each method fragment is labeled with one or many community size indicators, suggesting the most suitable community sizes.

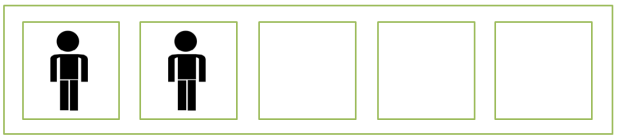

Figure 10 –Fragment Selection Indicator - Community size

6.02.2 Community Maturity

As shown in the vitality indicators sub-section (4.03), community age is a direct indicator for the probability of its survival and was found to have a significant positive correlation with the maturity of it software development processes and its governance structure. This in return makes it a suitable indicator for selecting applicable method fragment, as it can indicate which fragments are best suited for mature or young OSCs. Alike to the previous indicator, each fragment can have one or multiple labels describing matching OSC criteria.

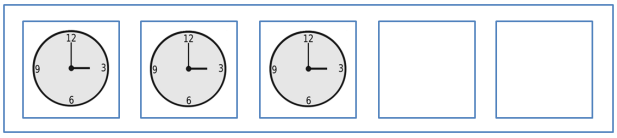

Figure 11 - Fragment Selection Indicator - Community maturity

6.02.3 Fragment Costs

Looking directly at the perspective of software product managers, disposable time and financial resources can vary greatly with each managed software product. Thus, even if a method fragment for vitality assessment is promising, it yet may not be feasible to use. This can be the case if the given fragment exceeds available project resources or if the overall analysis would benefit from focusing the efforts on less resource intensive fragments. In contrast to the previous indicator, each fragment is labeled with only one cost indicator that expresses the authors' estimate of the necessary financial or time investment, in order to execute the according method fragment.

Figure 12 - Fragment Selection Indicator – Fragment costs

6.03 Situational Method Fragments

By combining selection rationale and vitality indicators, this sub-section presents a number of OSC health method fragments that can be used individually or in combination by software product managers. Within the following sub sections, we present five fragment categories, each aimed at different, identified vitality indicators.

In regard to fragment completeness, It is noteworthy that three vitality indicators have not been included in the fragment pool. OSC license utilization, user interest and bug report frequency have been deliberately excluded from the method base. It is the authors' belief that these areas are either not relevant for software product managers or have been reviewed with contradictory results in scientific literature. As an example, OSCs with commercially not suitable licenses are unlikely to be considered for integration by software product managers.

Furthermore, the author would like to highlight that this method does not attempt to specifically dictate how to extract or quantify data but presents a guideline that can and should be adjusted to particular characteristics of each analyzed OSC.

6.03.1 Common Method Base

All OSC Health Analysis Methods contain two base fragments that act as a foundation for the entire vitality assessment. Figure 13 depicts the first fragment that focuses on data availability, as well as the second fragment that serves the purpose of establishing a base performance index for interval based method fragments (see Section 6.03.2).

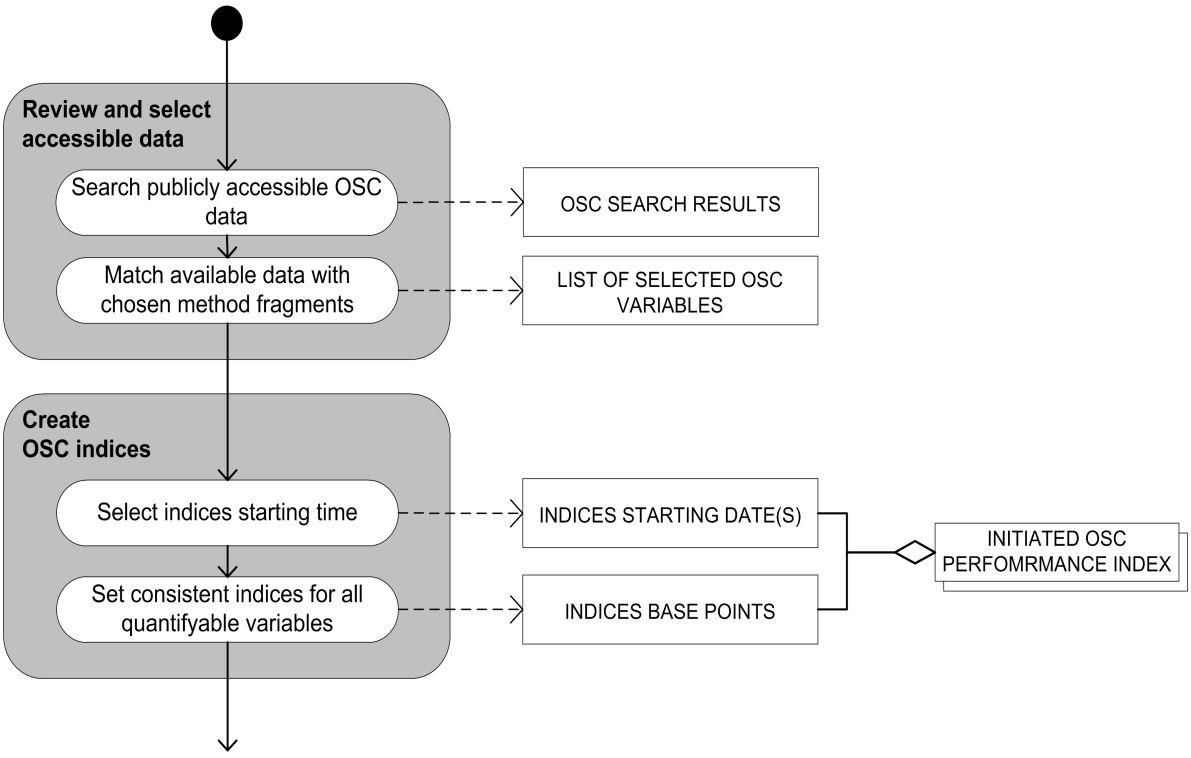

Figure 13 - OSC Health Method: Common method base

Both fragments can be seen as a prerequisite for structured data analysis and are thus not labeled with a selection rationale. Furthermore, the fragments should be seen as a mere guidance for adjusting the freely selected OSC fragments to a community's accessible data, as well as for standardizing data operationalization in order to achieve consistent and reproducible results across all index based fragments. This is achieved by calibrating each interval variable based on one common starting date and standardized base points. Alike to established common practices in stock market indices, this approach enables detailed performance tracking of each vitality variable over time. Furthermore, it enables the use of aggregate constructs for certain vitality aspects or for the entire OSC vitality.

6.03.2 Performance Index based Method Fragments

Looking at distinctly quantifiable vitality indicators, this sub-section introduces a number of method fragments that can be used to generate interval based vitality indices.

The first fragment, depicted in Figure 14, aims at external OSC metrics and is most suitable for measuring outward facing OSC activities. This includes actions, such as extracting the number of OSC downloads over time, tracking online search popularity or monitoring release frequency. All activities produce individual results, as well as an aggregate index, representing the overall metrics performance.

The fragment is suitable for a broad range of OSCs with varying degrees of maturity and size. However, due to the necessity to establish and track changes to the indices, the fragment is relatively expensive and has the second highest cost rating

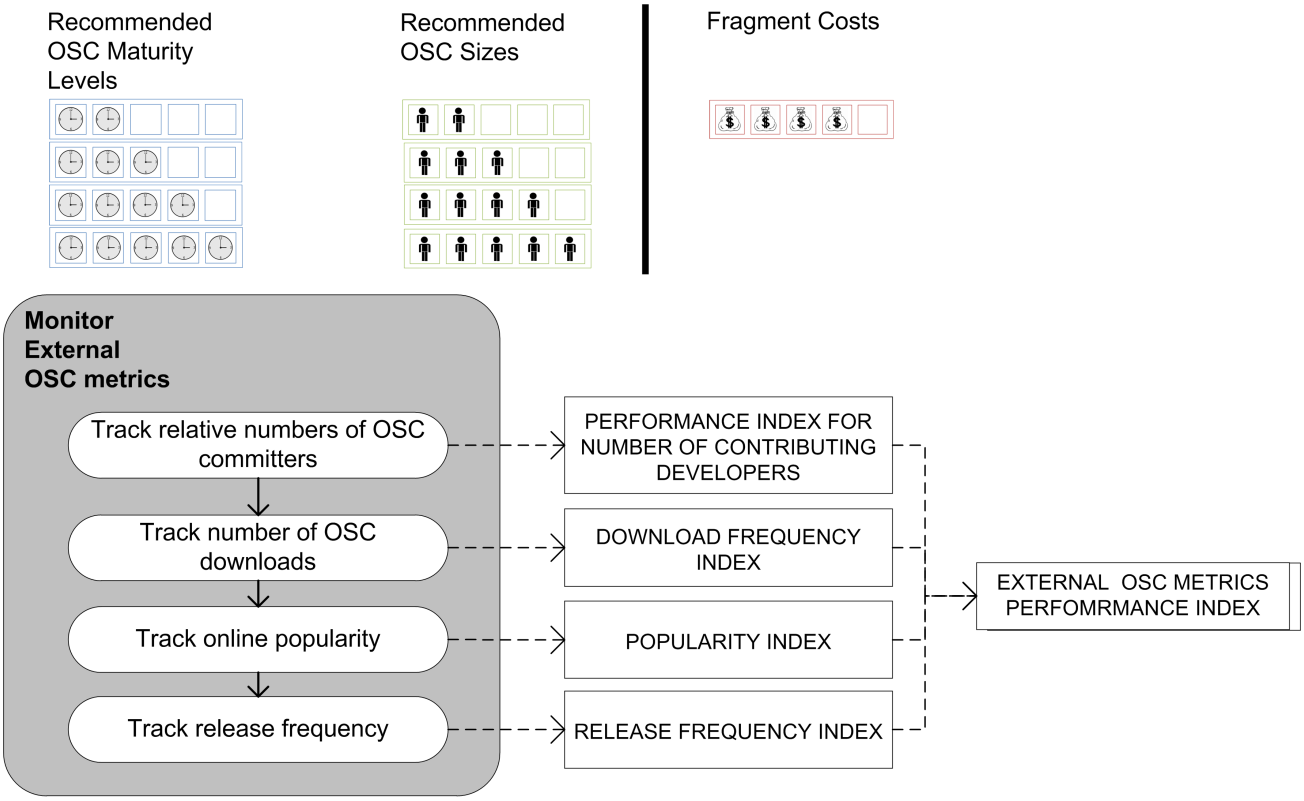

Figure 14- OSC Health Method Fragment: Monitor external metrics

In contrast to Figure 14, the following fragment (Figure 15) measures internal OSC activity and attempts to accurately depict internal community processes. Hereby, the fragment measures OSC aspects, such as bug or feedback response times, mailing list or forum activities and source code contributions. Analogue to the external metrics, each activity of this fragment produces a vitality index for the respective variable, which in return are aggregated into an internal OSC performance index. Due to their nature, this fragment is labeled with an identical selection rational to the one in Figure 14.

It is noteworthy that the indicated fragment costs for Figure 14 and Figure 15 are based on pessimistic assumptions regarding data availability. Initial experiments with OSC vitality analysis have shown that even among popular and well established communities, broad data availability should not be considered a given (see sub-section 6.05.1).

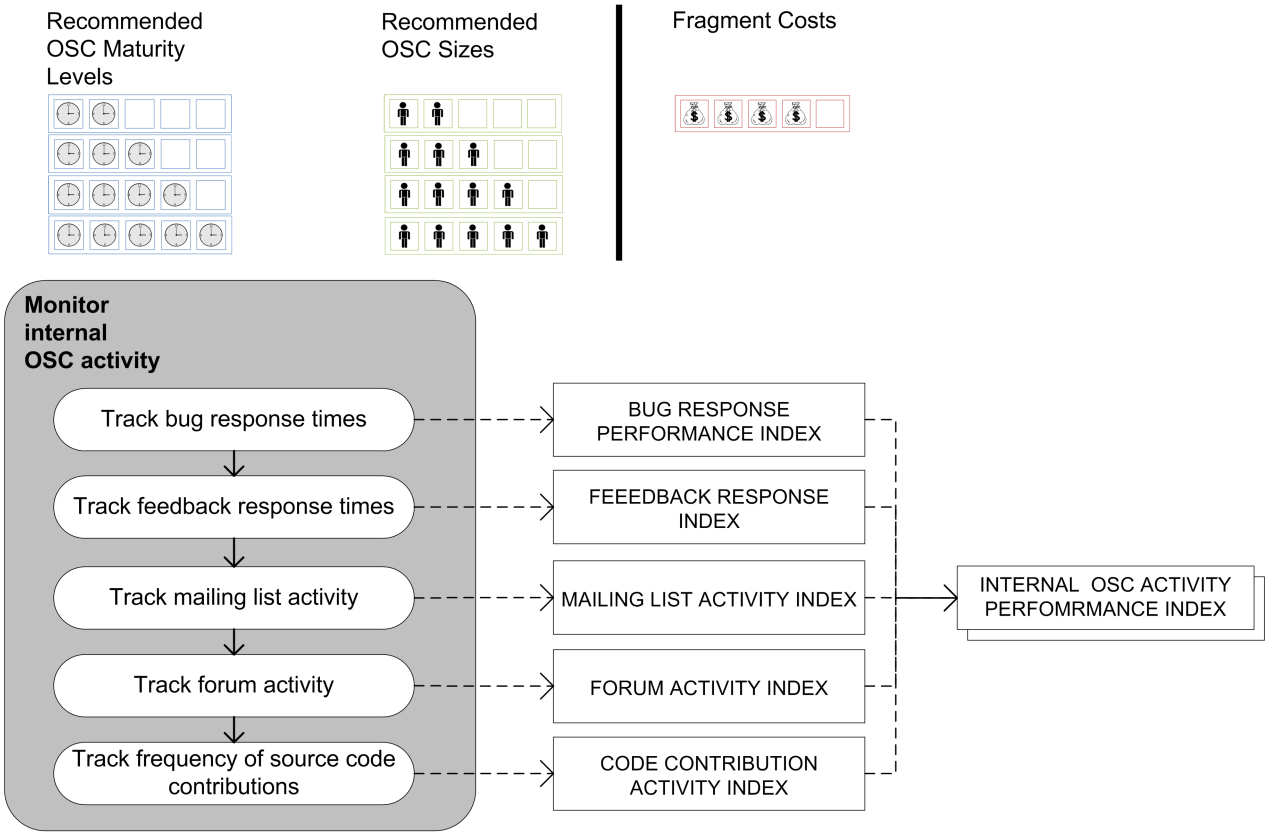

Figure 15- OSC Health Method Fragment: Monitor Internal community activity

The final index driven method fragment serves the purpose of correcting the generated indices for forms of cyclical community activity. It can correct too low or too high activity levels by taking a community's natural activity cycle into account and accordingly adjusting the individual and aggregate indices. Due to the necessity for historic data, as well as one for mature OSC processes, this fragment is only suited for medium to high OSC maturity levels. However, given that the indices solely need adjusting and no new data operationalization is necessary; this fragment is less expensive than its two predecessors.

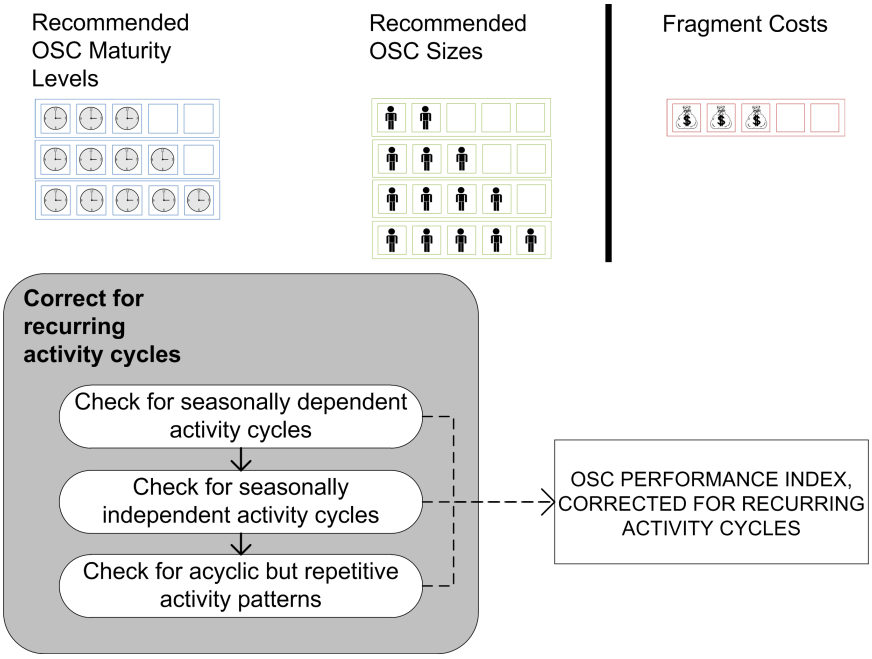

Figure 16 - OSC Health Method Fragment: Activity cycles

6.03.3 Scale Based Method Fragments

After having introduced the interval based vitality fragments; this section presents a number of ordinal scale Health Method fragments, based on vitality indicators that cannot be represented by means of a distinct unit scale. Instead, the indicators rely on a number of likert scales that are individually designed and adjusted for each fragment. Although this approach does not support a detailed monitoring of distinct variable changes, it allows for the tracking of proven indicators, which are otherwise difficult to operationalize, thus broadening the OSC analysis.

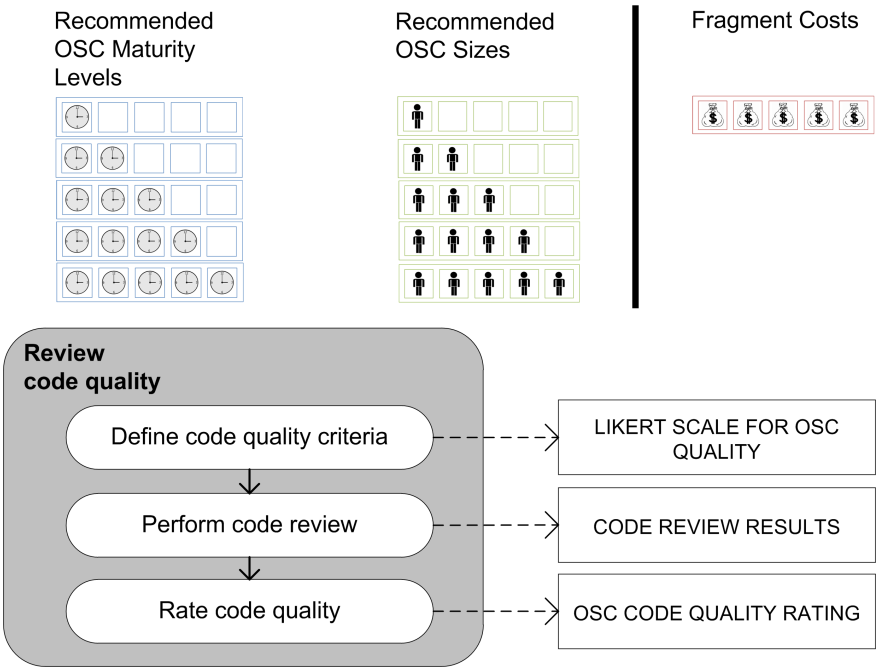

Figure 17- OSC Health Method Fragment: Source Code quality

Figure 17 evaluates OSC vitality from the perspective of code quality. After creating a definition of quality for the OSC in question, the method establishes a likert scale that indicates improvements or decline of code quality over time. Once reviewed, the results are being documented and the current state of the code quality rated. Given that code reviews are seen as good practice when integrating software components, the fragment is suitable for communities of virtually all sizes and degrees of maturity. However, it is noteworthy that manual code reviews are also very expensive, as considerable resources need to be invested, typically at the expense of development efforts.

Alike to code quality, Figure 19 depicts a method fragment aimed at evaluating the state of OSC project documentation. While only suited for medium to very mature OSCs, this approach is less resource intensive than the manual code review. After establishing a suitable scale for documentation quality, the results can be rated and re-evaluated at different points in time. Documentation quality was mentioned by a majority of the interviewed open source experts as a quick indicator for well-practiced community governance. Furthermore, it serves as an indicator for the necessary time investment when integrating an OSC.

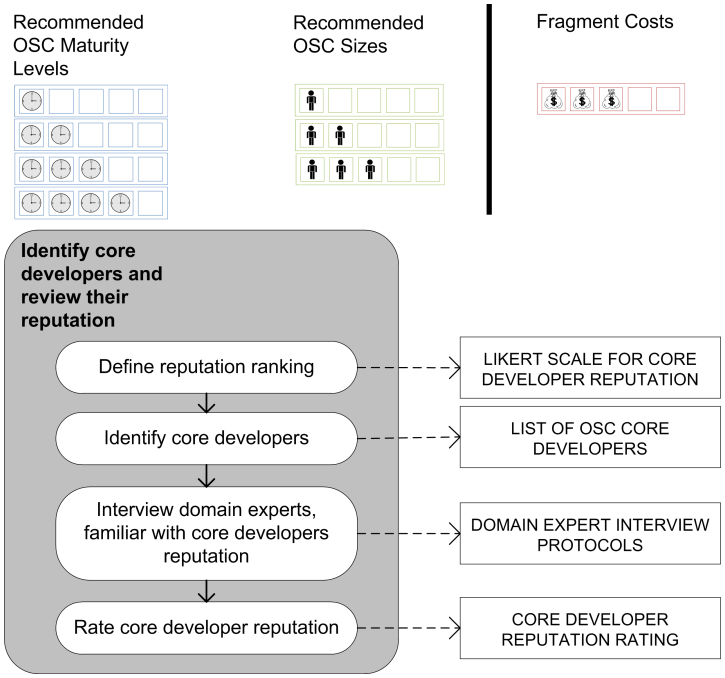

Figure 18- OSC Health Method Fragment: Core developer reputation

The purpose of the final two scale driven method fragments is to review OSC core developer reputation and motivation.

Analogue to the previous approaches, the fragment in Figure 18 first establishes a reputation ranking for core developers and then progresses to developer identification and conducting interviews with domain experts. Both steps are necessary for establishing a reputation ranking. As core developers play an important role in most OSCs, this fragment is suited for a broad scope of maturity levels, however becomes less significant for large and very large communities, as other vitality factors, such as community governance or a growing member base gain in significance. Due to its qualitative nature and the typically small number of core developers, this fragment only requires a medium time investment to be executed.

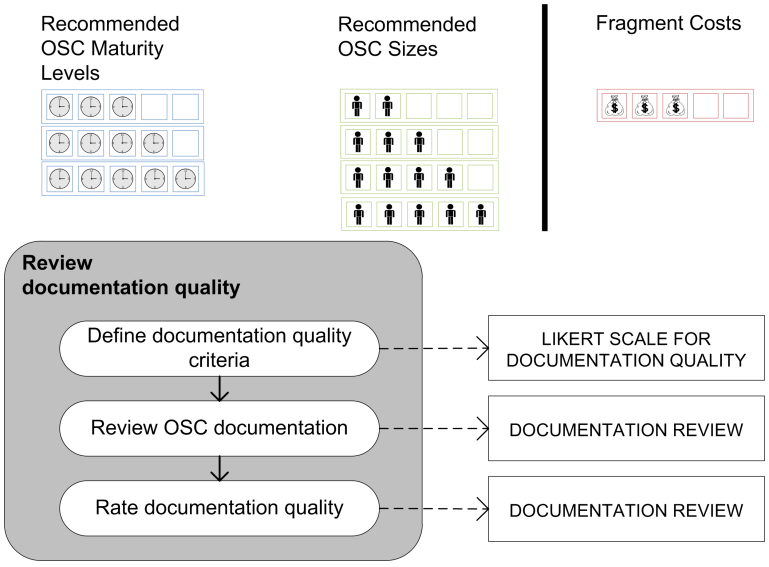

Figure 19- OSC Health Method Fragment: Documentation quality

The final fragment within this category (Figure 20) reviews one of the main drivers for OSC vitality: Core developer motivation. After clearly identifying core developers within a project's source code repositories or documentation, the fragment proposes the design of a core developer questionnaire and a matching motivation scale. Based on this, identified candidates can be interviewed and their motivation to contribute to the project tracked over time. As this approach is similar to one within the developer reputation fragment, the recommended OSC maturity and age indicators are identical. Hereby, the fragment costs are directly linked to the core developers' interest in cooperating with the product manager and should be reviewed in light of previous cooperation attempts.

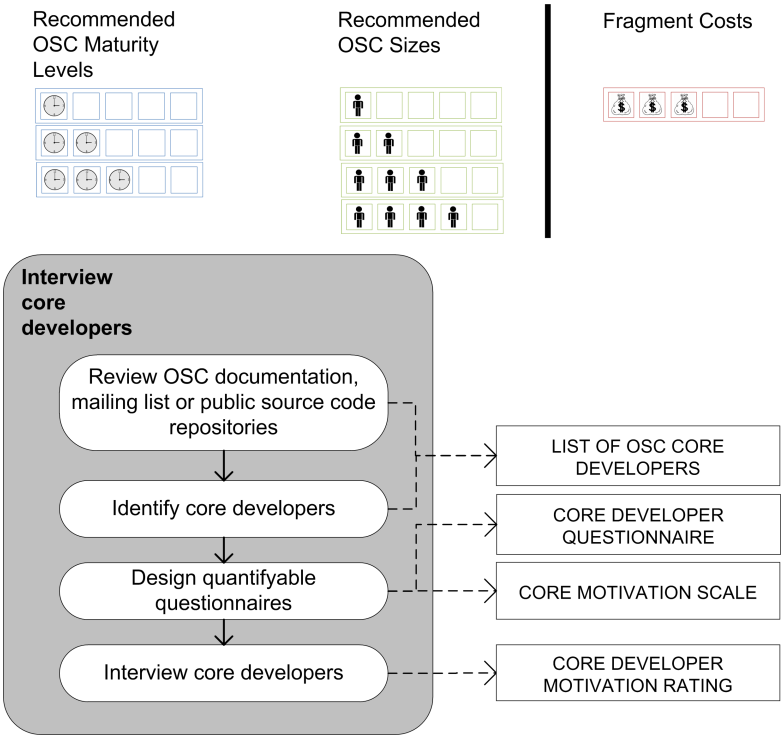

Figure 20 - OSC Health Method Fragment: Core Developer motivation

6.03.4 Software Development Practices Checklist

In contrast to the index or likert scale based method fragments, this subsection introduces a fragment that purely records the binary state of a number of vitality indicators within OSC software development practices. Frequently mentioned by both, the reviewed scientific literature and the interviewed OS experts, the software development processes within OSCs are a strong indicator for code quality and community vitality. Suitable for all, but the least mature and youngest communities, this fragment can quickly and without great time investment depict a community's stability. Additionally, communities that improve their development practices are also likely to grow in maturity and thus increase their probability of survival.

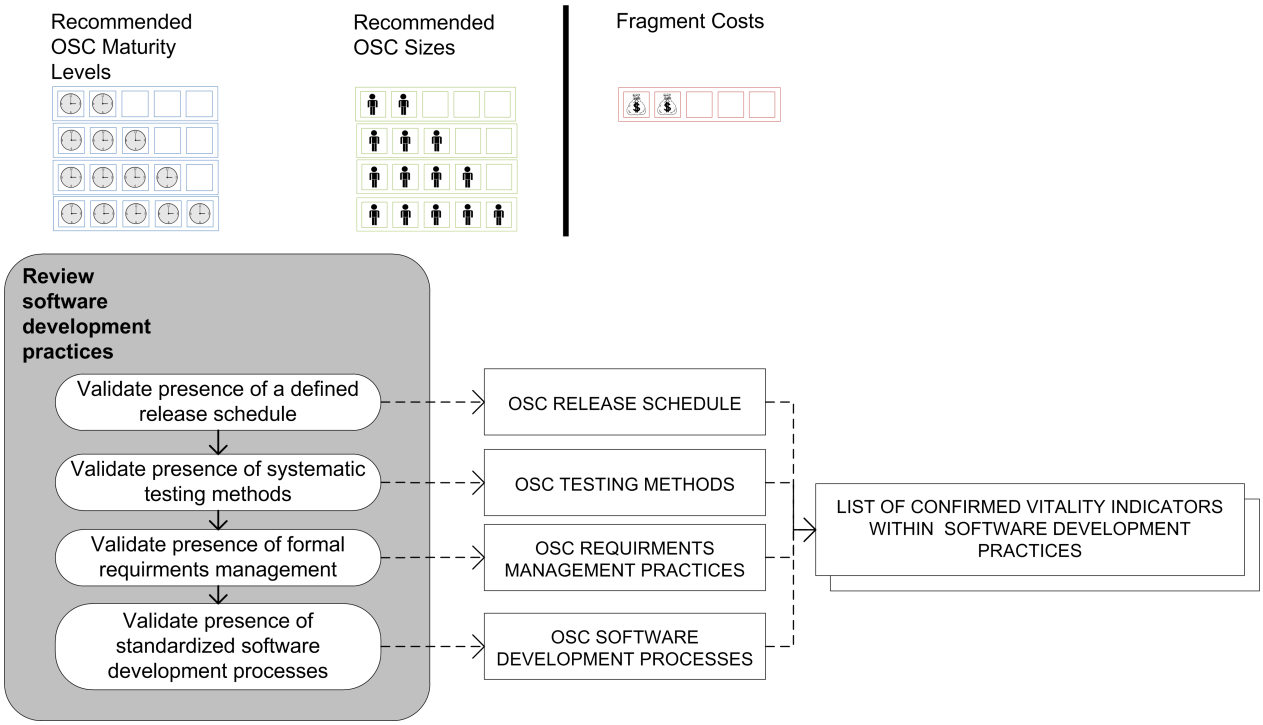

Figure 21- OSC Health Method Fragment: Software development practices

6.03.5 Alignment and Resource Optimization with Other OSC Stakeholders

The final fragment of the introduced fragment pool aims at reducing OSC monitoring costs by aligning business interests and dividing monitoring efforts among all commercial stakeholders sharing an interest in an OSC. Figure 22 illustrates an approach that systematically searches for fellow commercial stakeholders within a given OSC and attempts to identify mutually beneficial opportunities for all involved parties. Given the fact that open source is equally available to all software integrators, the authors assume that often identified commercial parties don't engage in direct competition over access to the OSC but are interested in its vitality and software quality. Thus, sharing monitoring costs can significantly contribute to reducing individual project manager workloads. Alternatively, such an approach can lead to a more accurate vitality analysis, as the increased cumulative resources allow for using a larger number of method fragments. Based on the fragment's nature, the authors find it suitable for all community sizes and maturity levels. Furthermore, given the fragment benefits, the resource investment is marginal and only involves search costs for identifying other stake holders and time investment for initial discussions.

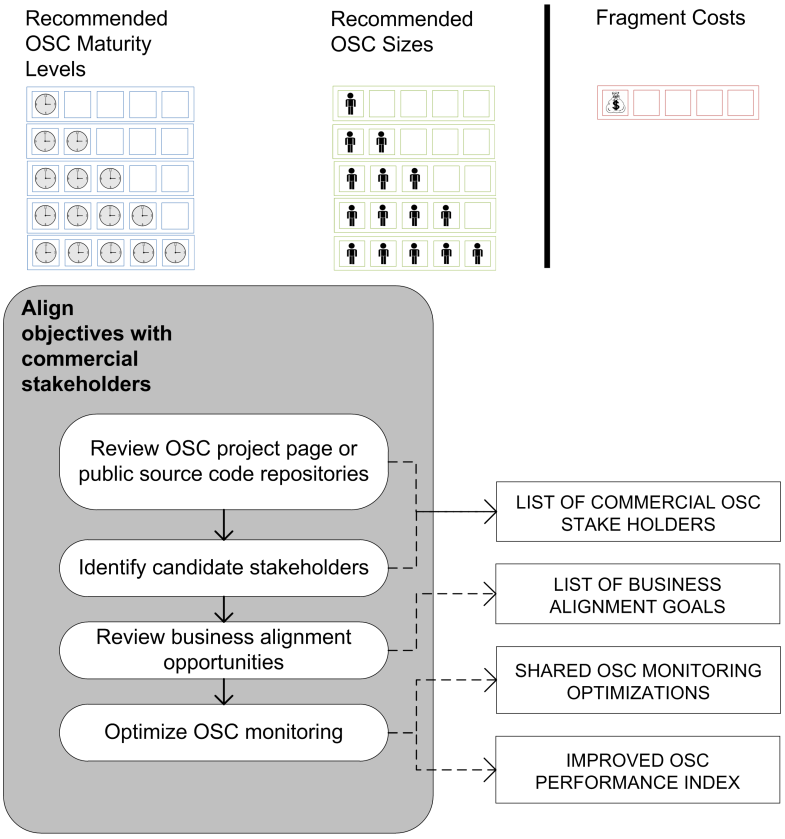

Figure 22 - OSC Health Method Fragment: Align goals with existing commercial stakeholders

6.04 Method Assembly and Calibration

Depending on fragment selection, the final situational method can contain any combination of index or scale based elements, as well as binary results. Additionally, two fragments with the purpose of either correcting activity indexes for cyclical trends or for optimizing OSC monitoring efforts can be included.

The only fragment required for method assembly is the method base. Consequently, any number or combination of fragments can be utilized within the situational method. Based on the chosen selection rationale, software product managers can group fragments and are free to either aggregate method results or compare them on an individual basis.

Specifically looking at index driven methods and the produced performance indices, they can be aggregated to create a unified OSC performance index. Next to the overall OSC health, individual vitality indicators can be monitored separately in order to identify trends within vitality areas. Furthermore, the approach supports index adjustment by correcting the individual weight of vitality indices according to their importance for a given OSC. Thus, a situational method cannot just be tuned by means of fragment selection but also by means of adjusting fragments to local OSC realities.

Method artifacts generated by non-index fragments lack a distinct scale; however they can be used to register changes in vitality. By individually tracking variations in the likert ratings, each method fragment can be used as an indicator for changes in OSC health. The same applies to the specific case of monitoring software development practices. Despite the binary nature of the results, each can be compared to past

practices of the OSC. Thus, added practices can be seen as an improvement to community health or code quality, while the abandonment of practices can be seen as a decline in community performance.

Although not necessary, fragment grouping can contribute to the ease of use of the method and provides it with a clear structure. For illustrative purposes, all presented method fragments have been assembled into a single situational method, covering all vitality aspects introduced within this thesis. This complete OSC Health Analysis method can be found in Appendix D.

6.05 Method Application

In order to present a practical example of the method's application, a representative OSC was selected for data analysis. Given the Health Analysis Method's situational nature and the great diversity among OSCs, the main reason for choosing a single OSC was to introduce a set of steps that can be taken in order to apply the introduced method fragments to a realistic assessment scenario. Furthermore, the aim was to present an analysis of a broadly known OSC and thus make the example as relatable as possible for project managers, familiarizing themselves with the new method.

OSC selection was performed based on the popularity index of one of the most widespread public source code repositories, namely Github. The platform hosts a vast amount of open source projects and provides additional features, such as social components, project collaboration tools, issue tracking or code reviews. Furthermore, Github is built around the distributed revision control system Git that provides an additional source of data. Due to Git's fully distributed architecture, revision data can be directly extracted from every cloned copy of a repository.

Among the highest rated community projects on Github, the jQuery JavaScript library was selected for analysis. Although ranked third in the index (8. May 2013), the first two projects Twitter Bootstrap and Node.js were excluded due to either their direct ties to a commercial entity or the scope of available data and community maturity. An additional reasons for excluding the two most popular projects was the authors' intentions to choose a typical OSC that had achieved a sufficient degree of maturity and had been exposed to a minimal degree of external influences affecting its development.

The selected OSC is a widely popular JavaScript library first introduced in 2006 by John Resing, a former developer at the Mozilla Foundation. The community's age, its data availability and its established mature development practices make the community ideal for the exemplified OSC vitality analysis.

The analysis within the following sections is based on community data available on Github, the jQery Git repository, Ohloh (a public directory of open source software owned by Black Duck Software, Inc.), the GHTorrent project, as well as community pages, blog entries, the mailing list and similar documentation of the jQery project. The jQery Git repository was mined with the open source tool GitStats[2].

[2] http://gitstats.sourceforge.net/

The following subsections contain illustrative data that can be used for OSC vitality analysis of the representative OSC. Hereby, the aspect of data availability will be discussed within the respective method sections.

6.05.1 Index based analysis

Looking at external OSC metrics, the relative number of OSC committers, community online popularity and release frequency can be derived from publicly available sources and are thus used for establishing a base performance index in the respective categories.

Figure 23 depicts the number of active contributors per month and thus illustrates changes in commit activity over time. Furthermore, the graph highlights the activity peaks around each major release (see Table 7) and shows the overall growth in activity when the project transitioned to a major release (v. 1.5) in 2010.

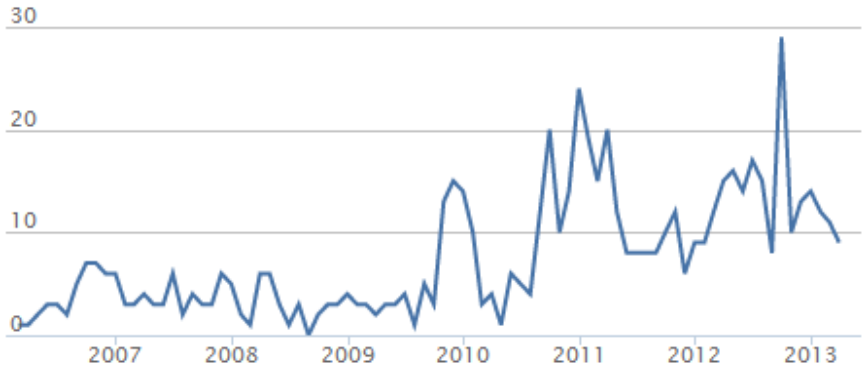

Figure 23 – Sample OSC: Number of unique contributors per month

Although the most recent Github API does offer the opportunity to extract the overall number of downloads, this data attribute is not tracked over time and does not take downloads from other mirrors into account. Unfortunately, the jQery foundation does not provide an aggregate overview of the total number of downloads for historic releases. Thus, this vitality indicator will not be included in this analysis.

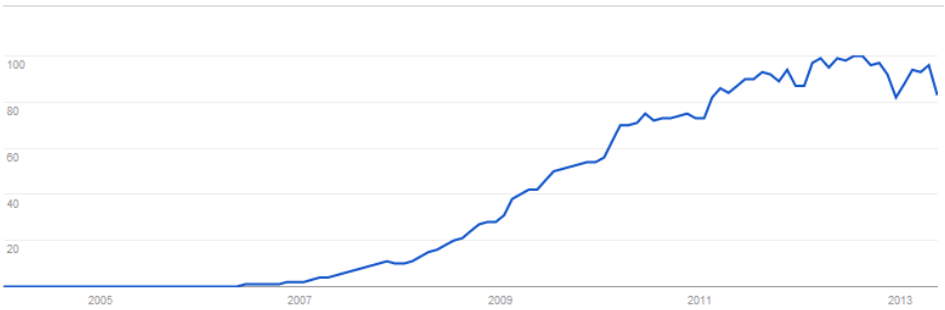

Figure 24 – Sample OSC: Online popularity in Google trending topics

A direct approach for assessing an OSC's online popularity is Google Trends. The tool depicts the overall trend for queries mentioning a particular word or topic. While the use of Google trends can be seen as problematic or potentially inaccurate with keywords that can also be found within unrelated expressions or sentences, this risk is unlikely to be relevant for the unique name of jQery. However, it is noteworthy that a

trend can be positive, as well as negative. Thus, a community that has received negative publicity or bad reviews could also appear as a positive search trend. Figure 24 depicts the trend performance for jQery. Hereby, the y- scale represents percent values with 100% set at the highest frequency of occurring searches.

The final vitality indicator within the external OSC metrics fragment is release frequency. Such information is available through a number of sources incl. Github or OS project trackers. For this analysis the data was directly extracted from the project's community blog. Table 7 depicts a list of all major software releases, as well as the elapsed time in month between each release.

Version number	Release date	Elapsed time in months after since release
2.0.0	18. April, 2013	2 months
1.9.1	04. February, 2013	1 month
1.9.0	15. January, 2013	6 months
1.8.0	09. August, 2012	9 months
1.7	03. November, 2011	6 months
1.6	03. May, 2011	4 months
1.5	31. January, 2011	12 months
1.4	14. January, 2010	12 months
1.3	14. January, 2009	4 months
1.2	10. September, 2007	8 months
1.1	14. January, 2007	5 months
1.0	26. August, 2006	

Table 7 – Sample OSC: Release frequency

As suggested by the method fragment, all measured indices are aggregated into one external performance index (Figure 25). For this purpose, each metric has been first recorded with the according activity for each month between Aug. 07 and Apr. 13. Due to the different measuring units, the performance was adjusted to changes in percent, relative to a standardized base value. The index for the number of active contributors per month is operationalized by changes in percent, compared to an average of 8 contributors per month. This value is based on the actual project average for its entire lifespan and has been rounded to an integer value. Release frequency is recorded in relative frequency compared to a release frequency of 2 releases per year or 0.16 releases per month. As online popularity is already measured in percent values, relative to an established base value, the index has been included in the analysis as is.

It is noteworthy, that such an approach is only suitable for a one-time analysis, as the 100 % mark of the popularity index is always adjusted to the number most frequently occurring searches in a community's history. Thus, should an OSC reach a new record in search frequency, historic data must be adjusted in order to maintain index consistency. Furthermore, the index is only designed to indicate trends and therefore uses average community performance for activity comparison. Although, minimum activity can also be used as a reference point for indicator performance, this was avoided in order to prevent a too striking upward trend on the graph that would make long term-trends less observable.

In addition to the regular aggregate index, two additional indices were added in order to illustrate the impact of weight adjustment of individual variables. The second index highlights the effects of a doubled weighting of the importance of the number of active contributors. The last index illustrates the effect of a reduced (half the influence) weight of online search popularity. All occurring spikes are either linked to activity related to a new version release (see Table 7) or a significant increase in release frequency. Especially the latter is noticeable for the time period of Oct. 12 and Feb. 13. Both spikes are primarily caused by a drastic change in release frequency to one and two months respectively. These releases occurred significantly more frequent than the average of only two new versions per year.

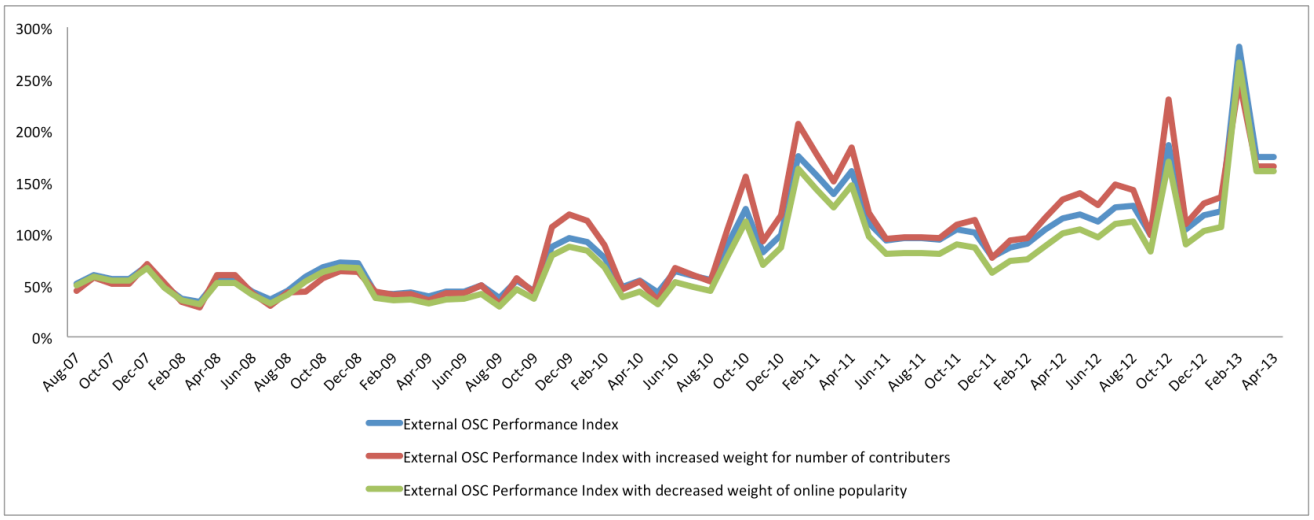

Figure 25 - Sample OSC: External performance index

In contrast to the external metrics, data availability for internal OSC performance indicators is limited. One of the reasons for the lack of consistent data is the age of the community. The OSC has undergone a number of hosting transitions for e.g. its forums and unlike the data available via Git repositories, does not provide consistent long term records. Furthermore, the community lacks a dedicated mailing list and often refers support questions to other platform, such as the stackoverflow [3] community, which covers a large number of jQuery related support discussions.

However, the jQuery foundation does list short term activity data for forum activity and ticket based issue tracking for a time span of the last week or month (see Appendix D). Therefore, suitable data could be accumulated by means of long term data mining.

The only vitality indicator covering the entire OSC life span, selected for evaluation, is the number of code contributions. Hereby, the indicator serves two purposes by both, highlighting the number and frequency of commits, as well as visualizing OSC activity cycles over a time span of 6 years (see Figure 26).

The figure highlights a seasonally cyclical activity pattern with peaks towards the end and the beginning of fife out of six years and low activity levels towards the summer periods. The only exception to the pattern occurred in summer 2012 when the community released a new library version outside its typical pattern.

[3] http://stackoverflow.com/

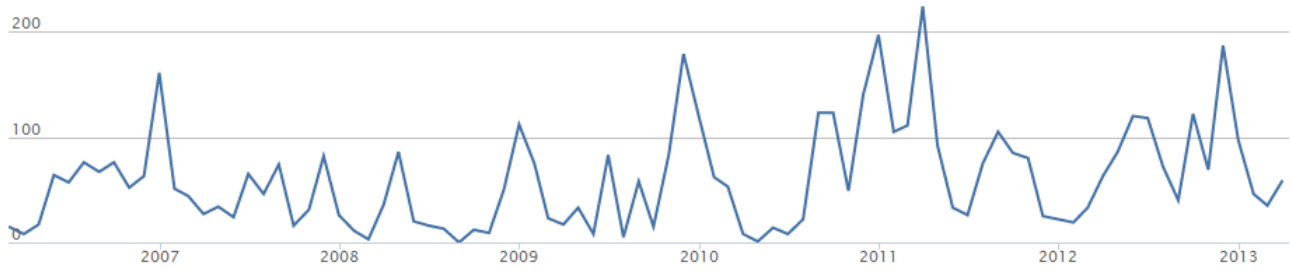

Figure 26 – Sample OSC: Activity cycle in commits per month

By combining the external performance indicators with the available internal data for commit frequency, an aggregate index can be generated that incorporates all long term community activity into a single Index. Figure 27 depicts a unified index that combines the external vitality indicator of Figure 25 with the OSC commit frequency introduced in Figure 26. Hereby, the commit frequency is expressed in a relative percent value based on an average commit frequency of 60 commits per month. Furthermore, all of the aggregated variables have been included with equal relative weight, as the internal OSC vitality indicators are only represented by one variable. Otherwise, commit frequency would have an unproportional influence on the aggregate index. In addition to the ordinary aggregate index, a second index that was corrected for peak activities in winter periods and low activities during summer time was added. The activity corrections apply to three winter and summer months, which have been reduced or increased by 20% in significance respectively.

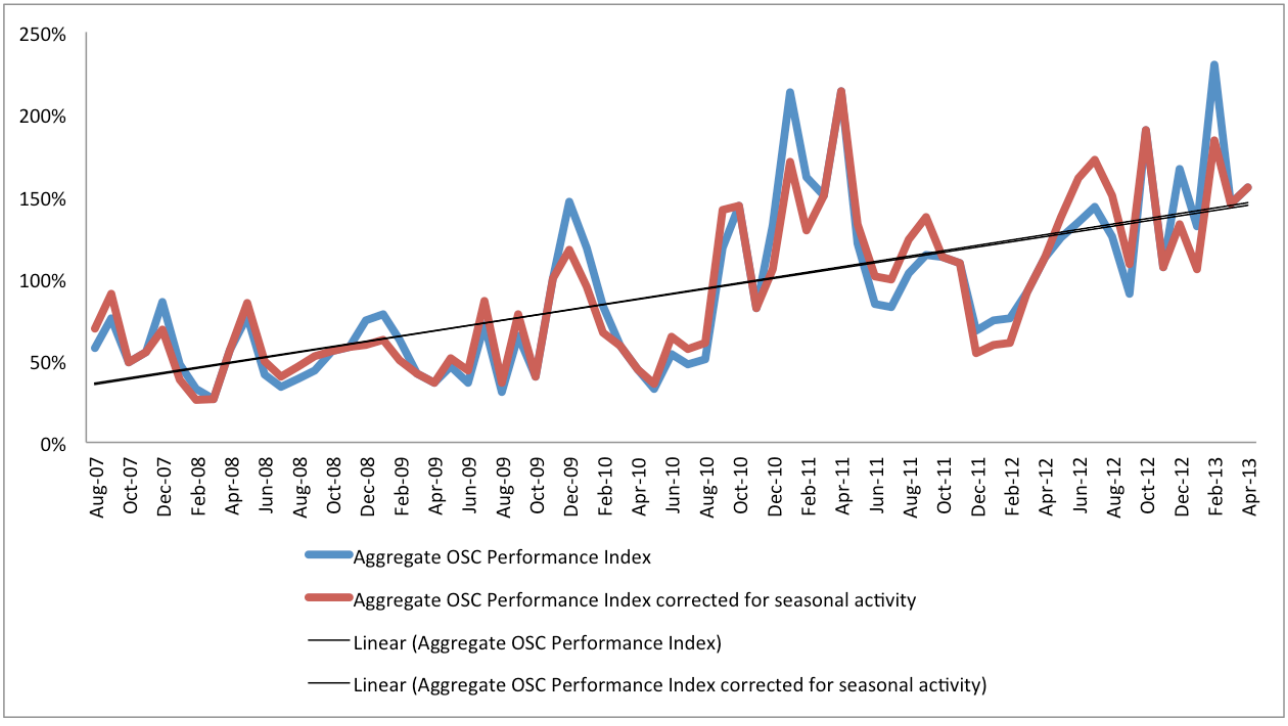

Figure 27 – Sample OSC: Aggregated performance Index and linear trend lines

It is noteworthy that when evaluating the graph, measurement consistency is most important while exact performance values lack significance. All index activity values are measured in relation to an established base value and could indicate stronger or weaker growth if the base is modified.

6.05.2 Questionnaires and Likert scales

Following the primarily data driven analysis of OSC activity, this section presents a more qualitative view on the sample OSC's practices and covers aspects of code and document quality, as well as core developer reputation. Given the illustrative nature of this example no community members were interviewed for OSC vitality assessment. Instead, the purpose of this sub-section is to introduce a number of techniques and likert scales that illustrate a possible application of the OSC Health Analysis method.

The first fragment within this category deals with OS code quality and requires pre-defined quality criteria in order to rate the component in question. While code quality ratings can be seen as controversial and heavily depend on the programming language, the reviewers' preferences or the application's purpose, generally acceptable criteria include the architecture of the application, DRY principles (don't repeat yourself), component modularization, coding practices for e.g. documentation or the use of best case software patterns.

In order to allow for simple comparison of changes in quality, a five level likert scale can be utilized. In the case of the Sample OSC, the scale can be ranked as follows:

1. Highly structured and optimized code
2. Well-structured code with little repetition
3. Moderately structured code without best practices
4. Manageable code structure with a number of complex dependencies
5. Low code quality with applying bad practices

Although a code review by a development team can bring valuable insights into a community's suitability for a given project, it is a highly time consuming and expensive approach. Alternative sources for OSC quality assessment can be aggregated online reviews, written by OSC users on platforms, such as ohloh.net. Due to the great popularity of the sample OSC, the library had received a total of 799 reviews (state 17. May), rating it with an average of 4.75 out of 5 stars. Applied to the above scale, this rating would rank the OSC's code as highly structured and optimized.

Analogue to code quality, the purpose of the documentation quality fragment is to create a likert scale that describes the state of an OSC's documentation and consequently to rate its performance over time. In contrast to code quality, documentation quality is less expensive to evaluate and requires little content specific expertise. Hereby, quality influencing factors can be the degree of detail, the presence of code examples, update frequency or ease of use and accessibility. A sample scale for documentation quality can be structured as follows:

1. Very well-structured, exhaustive and up to date documentation
2. Well-structured, exhaustive and mostly up to date documentation
3. Mostly well-structured but partially outdated documentation
4. Only partially structured or outdated documentation
5. Badly or unstructured documentation

In case of the example OSC, the authors rate the community with the highest quality level. It provides both, a very well-structured documentation that is fully up-to date, as well as an additional dedicated learning

center, designed to quickly familiarize new developers with the library's capabilities, as well as for introducing examples of new code features.

The third method fragment within the category deals with core developer reputation. Starting with a general ranking, the following scale can be used for assessing top OSC developers:

1. Widely recognized expert known beyond his/her domain of expertise
2. Recognized domain expert
3. Developer with acknowledged expertise (within the OSC)
4. Active contributor to the OSC
5. Less known developer

An exemplary approach for core developer identification is Git repository data mining. Indicators, such as the number or share of repository commits can be used as an approximation for the significance of a developer within the community. An analysis of the sample OSC revealed the ranking depicted in Table 8. Evidently, the most contributions to the OSC have been made by John Resig. Resig is widely considered the founder of the community and enjoys broad recognition within a number of technical domains, related to web technologies. Furthermore, Resig is a familiar actor within the broader open source community and counts the Mozilla Corporation as a former employer. Based on these observations, it is the authors' belief that he can be considered for the highest rating within the above scale. Although less known than Resig, other top contributors within the community enjoy a solid reputation and can be ranked within the second or third categories of the above likert scale.

Name	Number of commits (% of total contributions)	Number of lines of code added	Number of lines of code removed
John Resig	1714 (32.99%)	106571	92494
Dave Methvin	478 (9.20%)	6851	7260
Timmy Willison	404 (7.78%)	6034	4857
Julian Aubourg	330 (6.35%)	17875	15188
Jörn Zaefferer	327 (6.29%)	22681	21149
Rick Waldron	308 (5.93%)	6734	12142
Brandon Aaron	250 (4.81%)	11102	5408
Ariel Flesler	200 (3.85%)	18053	3317
Richard Gibson	138 (2.66%)	5401	4751
Oleg Gaidarenko	104 (2.00%)	1927	1387

Table 8 – Sample OSC: Core developers ranked my number of commits

The final proposed method fragment of this sub-section deals with core developer motivation. It suggests to identify community leaders, design a questionnaire to assess their motivation for contributing to the community and consequently perform interviews to acquire the necessary data. The approach for core

developer identification overlaps with the activities of the core developer reputation review and thus does not need to be repeated at this step. The results with the leading OSC contributors are depicted in Table 8.

Given a five-stage likert scale, ranking from “does not apply” to “fully applies”, a set of suitable interview question can be introduced, asking to what degree the following statements apply:

- I am motivated to contribute to the community.
- I have sufficient time to participate in community activities and to invest development time.
- I perceive the contribution to the community as rewarding and feel appreciated
- I feel that the community is moving forward in a good direction.
- I feel that I am being supported by fellow OSC developers.
- I feel that I am being supported by all community members.
- I feel that the created software is valuable by the community.
- I feel that my participation in the community is improving my professional skills.

Hereby, the purpose of the above questions is to assess the core developer’s motivation to further lead the project. Ideally the measurement should be repeated over time in order to identify on positive or negative trends within the core team. The results and implications for the entire OSC can be reviewed in light of the dependencies of the community interaction model, introduced in section 4.02.

6.05.3 Community practices

Given the age and acquired level of maturity, the sample OSC follows all of the suggested software development practices. In particular a number of development processes have been consistently standardized across jQery core, as well as other jQuery projects, such as jQery UI or jQery Mobile. An overview of the evaluated activities is depicted in Table 9.

OSC Development Practice	Presence
Defined Release Schedule	✓
Systematic Testing Methods	✓
Formal Requirements Management	✓
Standardized Software Development Processes	✓

Table 9 - Sample OSC: Software development practices in place

6.05.4 Suitable partners for business alignment

The final fragment of the OSC Health Analysis Method does not deal directly with vitality assessment but with business alignment with other stakeholder, sharing an interest in the same OSC. The purpose of the associated activities is to optimize resource allocation for vitality assessment, which can either be used to reduce monitoring costs or to increase measurement accuracy or scope.

Given the domain of the sample OSC, the primary target groups for monitoring alignment are software developers with a focus on web development solutions. This can include providers of content management systems, web-framework, as well as front-end developers. Alternatively, smaller organizations that strongly depend on the library can also be considered suitable partners for shared monitoring activities.

In the case of the sample OSC, it is particularly easy to find suitable candidate organizations. The jQery foundation, which is the umbrella organization for jQuery core and other jQuery based OS projects, publicly lists corporate foundation members. Among the listed gold, silver and bronze partners a number of non-profit, as well as commercial organizations can be found. All of them can be considered strong supporters of jQuery.

7. Validation

The following chapter contains the evaluation results for the OSC Health Analysis Method, introduced in chapter 6. The first sub-section presents general method suitability aspects, such as the method's applicability to the domain of software product management or the interviewees' motivation for applying a more structured approach than the current practices. The second sub-section deals with method usability aspects, such as fragment selection, assembly and execution. Focusing on the interviewees' experiences with assembling situational OSC Health Analysis methods, it illustrates the applicability of the method in a simulated environment.

7.01 General Method Suitability

In regard to the general value of applying a structured approach to OSC vitality analysis, virtually all interviewees stated that a more formal approach is preferred over the current individual ad-hoc analysis. Furthermore, seven interviewees stated that the introduced research domain and vitality analysis method raised their awareness for open source related risks and benefits.

A majority of the interviewees stated that the presented method exceeds the current practices and rated the additional degree of information a valuable contribution to overall product quality. Three interviewees raised their concern that the method would be best placed at the level of the current Open Source Core Team (see Section 5.01.1) and should not be applied on local level. However, all objections were explicitly limited to the framework of the current open source related processes of the case company. Despite the IBM specific concerns, all interviewees highlighted that, in general, the presented method can be considered a valuable extension to software product management practices.

Specifically looking at method usability, eight out of ten interviewees stated that they would be interested in applying the presented method in future open source related activities. Two interviewees stated that the method, while potentially more accurate than their current approach, would probably consume too much of their time and that they would prefer to rely on a more standardized tool or rating platform. It is noteworthy that all time related objections were made by interviewees whose primary activities are based on custom projects that only utilize open source for short term customer engagements, without the long term liability risks of product software.

Among those interviewees indicating their unreserved approval for the proposed method, all mentioned a preference for adjusting their OSC related time investment on a case by case basis. Particularly, five interviewees who were already following an established or self-developed pattern for OSC analysis stated their interest in testing the method's capabilities in contrast to their usual assessment approach.

The two interviewees declaring their preference for a tool based analysis further elaborated on their view on open source vitality. In their opinion, an ideal OSC analysis is best monitored by an external, neutral party and not by individual product managers. Hereby, the example of research firms, such as Gartner or rating agencies, such as Moody's were brought up, both highlighted that individual and potentially redundant monitoring of OSCs may not be economical.

Specifically asked about the value and suitability of the presented vitality indicators, all interviewees agreed with the presented list and no objections or remarks were made. Furthermore, all interviewers stated their approval for the scientifically evaluated indicators. Particularly those interviewees already practicing OSC analysis, showed interest in the origin and potential applicability of indicators stemming from scientific research (See Section 4.03).

7.02 Method Applicability

Following a general evaluation of the method's suitability for vitality analysis, this section presents the interviewees responses to different aspects of the OSC Health Analysis Method. After having completed the task of assembling three situational methods for sample OSCs, the interviewees were asked to describe their experiences with the process. Hereby, the introduced scenarios covered the perspectives of a small IT service provider, a small or medium enterprise (SME), as well as a large product software developer. The next four sub-sections introduce the categorized interviewee responses.

7.02.1 Fragment Selection

For all three cases, fragment selection has been found to be clear and transparent. Specifically the three categories for the selection rationale were perceived positively by all interviewees. Hereby, eight interviewees mentioned a particular preference for the fragment cost indicator, followed by the recommended community sizes. Community maturity was generally understood, however it required some additional clarification during half of the interviews. This observation suggests that a better term or an alternative indicator might be more suitable for this purpose.

Looking at the presented fragment categories, four interviewees required further explanation for understanding the difference between the index and scale based fragments. Fragments aimed at business alignment or reviews of current practices were perceived without any difficulties.

Four interviewees stated that they had preferred to have read a written manual or documentation to deepen their understanding of each fragment's purpose and categorization. The same interviewees mentioned that having seen an example of the method's application on a real OSC, before attempting to implement it during the interview, would have likely improved their performance.

Faced with the three selection scenarios, virtually all interviewees first reviewed the fragment indicators, before dealing with the fragments' actual purpose. The scenarios describing the small IT service provider and the SME required a greater degree of resource optimization and required more time for completion than the case describing a larger software vendor. Especially, when designing the situational method for the scenario with the smallest scale, the interviewees compared many fragments with each other, contrary to just excluding not suitable ones. Nine interviewees came to a small selection of three to five, mostly overlapping, and least resource intensive fragments. Virtually all interviewees stated that this scenario was the most difficult one, as it required them to wage a number of hypothetical factors without being familiar with an actual OSC.

The SME scenario was perceived as less difficult and fragment selection required less time than the first two cases. The only two challenges were reported in regard to index driven fragments and business alignment. Five interviewees felt that they did not have sufficient information about the presented OSC to make assumptions about data availability for index based fragments. Four interviewees were uncertain

about the likelihood of finding other stakeholders in the OSC that would be willing to share and coordinate monitoring efforts.

The last and as least challenging perceived scenario was the case with the greatest available project resources. Without the perceived need to optimize fragment selection, nine interviewees choose all available fragments. One interview deliberately excluded the most resource intensive fragments and stated that the necessary time investment can be divided across a number of suitable monitoring partners, also relying on the OSC in question.

7.02.2 Fragment Assembly

In contrast to fragment selection, fragment assembly was completed by all interviewees in a considerably speedier manner. Particularly those cases requiring a limited selection of method fragments were assembled within less than two minutes. All interviewees stated that this step was significantly easier to execute than the selection process. Four interviewees attempted to categorize their fragments into logical execution groups. The remaining interviewees simply aligned the selected fragments beneath each other in a random order. Two interviewees raised concern regarding the optimal order for execution efficiency while the remaining eight did not mention any issues at all.

7.02.3 Fragment Calibration

All interviewees acknowledged the benefits of adjusting fragments to the situational requirements of each monitored OSCs. Hereby, a number of concerns in regard to the accuracy of fragment measurements were raised. Index driven fragments were found to be most difficult to compare by six interviewees. All six mentioned that custom adjustments by one product manager might not be comparable to the results of another one, reviewing the same OSC. Concerns about calibration of index driven fragments were mentioned by four interviewees. In their opinion each interview was potentially exposed to the interviewer's bias and additional adjustment to a questionnaire could potentially jeopardize the results' comparability with data gathered during previous interviews.

7.02.4 Fragment Execution

Asked if the interviewees would see themselves able to execute the assembled OSC Health Analysis methods, all confirmed their general capability to do so, however raised a number of concerns, which are discussed within this sub-section.

Due to the hypothetical nature of the presented scenarios and the lack of familiarity with an actual OSC, seven interviewees mentioned concerns regarding sufficient availability of data to perform the analysis. Without having seen a sample execution of the method, they were uncertain about the necessary tasks in order to perform data mining, data analysis or index construction. Five interviewees mentioned that they would prefer a number of applicability examples, prior to investing too much time into attempting to mine and analyze data themselves.

Three interviewees issued their concerns about data maturity and the lack of predictability of trends for young OSCs that have not generated enough data, necessary to identify growth patterns. Asked about the recommended minimum age for an OSC, the interviewees suggested an average of one and a half to two years. Further concerns were raised about trend accuracy. Due to the situational nature of each fragment, it is possible that the individual bias of a software product manager will affect his choices, thus making his

results not generalizable and decrease their value for other OSC observers. Furthermore, a strong bias could negatively affect potential OSC monitoring alignment with other parties.

Reviewing the interviewees' feedback on the scale based method fragments, the category brought up less concerns than the index driven one. All comments and remarks revolved around the accessibility of interview partners and questions regarding potential pitfalls of interview driven data operationalization.

Four interviewees stated that they found it unlikely that core developer of OSCs would be willing to repeatedly participate in interviews, initiated by a commercial party. Three other OS experts mentioned that it might prove difficult to convert qualitative feedback into a variable that is measured over time. It is noteworthy that those remarks were drawn back, after a sample likert scale with standardized questions was shown as an example for a possible feedback operationalization.

The checklist method fragment, dealing with OSC software development practices, was fully acknowledged by all interviewees without any reservations. The final fragment for business alignment with other commercial stakeholders caused a number of discussions regarding the search of suitable partners. Four interviewees stated that they were skeptical about finding partners for small or young communities. However, the same interviewees also mentioned that this should not be an obstacle for larger or more established OS projects.

8. Discussion & Conclusion

The following chapter sums up the final conclusions and results of the conducted open source vitality research. Specifically, it introduces a general overview of the findings and research contributions of this thesis, links the research results to the research questions and introduces general research limitations. Furthermore, a discussion of the OSC Health Analysis Method's applicability and its limitations, recommendations for future research, as well as final general conclusions are presented.

8.01 Findings and Research Contribution

This research provides a substantial contribution to the body of knowledge on open source vitality analysis and can be used to improve both, the scientific understanding of open source community vitality, community member interactions, as well as industry practices for software product management with open source dependencies. Furthermore, the conducted systematic literature review on OS vitality, the review of current OS practices of a major software product firm, as well as expert interviews with IT professionals with open source expertise are contributions to a broader understanding of the complexities and benefits of open source utilization in commercial software products. Additional research contributions are the introduced open source community interaction model, as well as the list of scientifically and empirically validated vitality indicators for OSCs.

In regard to research findings, the most noteworthy discovery was the rise of commercially oriented parties within OS ecosystems. Throughout the research interviews and the design phase of the OSC Health Analysis method, a number of organizations had been discovered that were playing an active role within what can be seen as a broader software ecosystem, comprised of commercially oriented software firms, as well as voluntary members of open source communities. Furthermore, many contemporary participation models appear to dilute the classical distinction between voluntary private contributions to open source communities and software development for commercial software firms. The introduced community interaction model (sub-section 4.02) and the review of corporate participation in open source (sub-section 4.04) attempt to model and further describe these new relationships within the existing ecosystems.

Specifically looking at contemporary scientific literature on the emerging relationships between open source and commercial entities, the conducted literature review revealed an abundant lack of scientific papers addressing this topic.

8.02 General Conclusion

Analyzing open source vitality, this research introduces an Open Source Community (OSC) Health Analysis Method, specifically aimed at the needs of software product managers, handling open source components within their managed products. The introduced approach allows product managers to create custom

vitality analysis methods that are adjusted to community characteristics and take available time and resources into consideration.

The method is based on a systematic literature review (section 3.03), an analysis of open source participators and a community interaction model (section 4), as well as an analysis of open source related practices at a representative case company (section 5).

Eighteen validated vitality indicators (section 4.03) and three selection indicators (section 6.02) lay the foundation for a pool of method fragments (section 6.01) that serves as the base for situational method design for OSC vitality analysis.

The developed method is validated by means of semi structured interviews, as well as exercises in method assembly by open source experts at IBM BeLux (section 7). The validation results confirm the method's general applicability for structured vitality analysis, as well as the utilized vitality indicators.

8.03 Thesis Results Linked to Research Questions

The purpose of this sub-section is to match the main and sub research questions, introduced within the first chapter of this thesis, with a list of brief answers that refer to the corresponding chapters, containing a more elaborate explanation.

The main research question of the thesis is as follows:

"How can Software Product Managers systematically and strategically evaluate the health and implementation of open source components within their commercial software products?"

The main deliverable of this research is the OSC Health Analysis Method (section 6). It is a modular method, specifically designed to be adjustable to situational requirements of varying OSCs and project characteristics. The method is centered on a pool of method fragments, each aimed at validated OSC vitality indicators and labeled with an easy selection rational. All fragments can be freely assembled and support a broad range of different data types that can either be processed individually or aggregated into an overall performance index (see section 6.05).

The thesis' results for the sub-research questions are presented below:

Which are the relevant vitality indicators for community health?

OSC vitality indicators are gathered from scientific literature and expert interviews, conducted at the facilitating case company. A list with all 18 identified indicators and their respective origins can be found in section 4.03

Is community health measurable and comparable by means of an evaluation method?

This sub-question was partially confirmed during the validation interviews for the OSC Health Analysis Method. All interviewees stated that the introduced OSC Health Analysis Method was indeed a suitable and more structured approach for vitality analysis than their current practices (see section 7.02). However,

vitality comparability can proof difficult, as OSCs are often very diverse. Furthermore, the situational nature of the introduced vitality assessment method makes direct comparability impractical. Each software product manager can adjust the method's fragment selection and data operationalization to his personal needs. This flexibility further complicates objective vitality comparison.

Which are suitable analysis techniques to demonstrate changes in community activities over time?

The OSC Health Analysis method categorizes method fragments, which are directly aimed at monitoring changes in vitality. The vitality centered segments of the method are comprised of the following three categories: Index driven fragments, scale driven fragments, as well as a checklist for OSC practices (section 6.03). All three are used to illustrate vitality changes. Hereby, due to the nature of the operationalized data, the index driven fragments are best suited to highlight changes over time and can display individual trends, as well as aggregate performance indices. The least suitable fragment is based on binary data of community practices and can only indicate change by showing that either a new practice has been adopted or an existing practice dropped.

Which measures can software product managers take to respond to changes in the health of open source communities that their product depends on?

This sub-research question was primarily addressed by the community interaction model, introduced in section 4.02. Hereby, the primary measures that have been found to positively influence community activity are: Financial and legal support, active community involvement, as well as direct code contributions. In the special case of open source projects that have been founded by commercial software firms, ensuring project accessibility to external developers has been found to positively affect overall community activity.

8.04 Research Limitations

Three research limitations are particularly noteworthy and need to be kept in mind when reviewing the research results.

The individual vitality indicators have been thoroughly validated; however their implementation within the OSC Health Analysis Method is based on the authors' understanding of open source vitality and the needs of software product managers. Only further research and additional case studies or experiments can proof the suitability of the chosen method engineering approach.

Open source communities are often part of entire networks of communities that integrate components or libraries into larger software constructs. As an example, various Linux flavored operating systems are based on software components which are written in individual communities that provide specific functionalities, such as the graphical user interface. The OSC Health Analysis Method exclusively looks at one OSC at a time and does not review the influence of components stemming from other open source projects. Although it would be possible to apply the method to all OS based dependencies, such an approach is likely to be very resource intensive and could greatly complicate vitality analysis.

The introduced exemplary applicability of the OSC Health Analysis Method is not based on an actual case of OSC integration but was solely included as a reference to demonstrate one possible way of performing vitality analysis on a popular and widely known OSC. The way the method will be applied and reviewed by SPMs can be heavily influenced by the included examples. However, it is not possible to just include the method fragments without illustrating a possible applicability as most interviewed OS experts mentioned the need to review an example of the applied method, prior to using it themselves.

8.05 OSC Health Analysis Method Applicability & Limitations

Specifically looking at the applicability of the method, virtually all interviewed OS experts confirmed that the method could be a valuable extension to their practices and therefore does fulfill its purpose. Especially the method's flexibility and the simple selection rationale for method fragments were highlighted as great benefits of the proposed approach.

Potential downsides to the introduced flexibility are the lack of consistency or comparability of results, generated by different SPMs. Due to the provided freedom to choose suitable fragments and to the capability of adjusting measurements to available data, the comparability of the vitality analysis results can proof difficult. Vitality for the same OSC can only be objectively compared if all participating SPMs standardize their implementation of OSC Health Analysis Methods in advance.

An additional limitation can arise from a lack of available OSC data. As illustrated with the presented example in section 6.05.1, a lack of data can be an obstacle to a complete vitality analysis. Next to general data availability, young OSCs can exhibit great activity spikes, which can complicate vitality assessment. In the case of the sample OSC, the activity of the first six month was characterized by striking growth mostly due to a quickly expanding code base. However, after an initial steep growth period the community stabilized at a more consistent and slower growth rate. Therefore, the method might be more suitable for communities that are old enough to have generated sufficient data for making assumptions about vitality trends within the OSC.

Specifically looking at method refinement, it has been viewed as a suitable approach for adjusting vitality measurements to an OSC's characteristics. However, a number of interviewees raised concerns that it is not efficient to place it on an individual SPM level. Instead recommendations were made to extend the concept of the method fragment pool to the organizational level and have each SPM's adjustments affect the available fragments for future assessment by other SPMs or monitoring partners.

A final noteworthy limitation of the proposed vitality assessment is the method's indifference towards OSC development strategies or roadmaps. This implies that while a community might be perfectly healthy and consistently growing, the architectural decisions it makes can have a negative impact on software products depending on it. Such changes can lead to situations when commercial software developers can no longer simply update to the newest version of a given component, but must adjust their application to match the new OSC architecture. The resulting and unforeseen efforts could exceed the costs of transitioning to a new library or internalizing the desired feature.

8.06 Future research

Throughout the method design and interview phases a number of interesting opportunities for future research emerged. All of the following recommendations are either based on current limitations of the research design and operationalization or represent an extension to the existing research scope and perspective.

One of the first and the probably most striking observation of this thesis is the current lack of open source research with a focus on commercially oriented organizations. The review of the case company's OS related strategic goals and the evident involvement of commercial entities in numerous popular open source projects encourage further research into this emerging form of hybrid software ecosystems.

Furthermore, a dedicated case study in larger software firms could reveal the optimal utilization of an OSC method fragment pool that can be used across a number of different software products or shared between organizations reviewing similar OSCs.

Focusing on a full ecosystem scope, a vitality analysis of individual communities, as well as a review of the aggregate ecosystem-vitality can extend the existing research towards a holistic OS health evaluation. Suitable ecosystems for such research could be a number of Linux distributions or the broader Eclipse community.

Additional research could be aimed at the optimization of monitoring costs by focusing on the role of information brokers, such as Black Duck's OHLOH directory. New insights could be gained from reviewing the value of a new organization within the ecosystem that could take on the role of a professional analyst of OSCs. Examples for related rating institutions already exist within the financial industry.

Finally, the introduced vitality assessment method would benefit from further validation, based on case studies with suitable OSC integrators. A review of old practices and a consequent comparison of measurements performed with the OSC Health Analysis Method could provide additional insights into the method's suitability and applicability in a professional environment. Performed at different case companies of varying sizes, a thorough review could establish the suitability of the method for small companies that are lacking monitoring resources. Furthermore, a larger set of case studies could allow for studying potential complexities for monitoring alignment of the same OSC across different organizations.

9. Bibliography

Barbosa, O., & Alves, C. (2011). A Systematic Mapping Study on Software Ecosystems. *Ecosystems*, *2*(4), 15–26.

Berander, P. (2007). Evolving Prioritization for Software Product Management. *Engineering*, 250. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.97.9698

Bessen, J. E. (2001). Open Source Software: Free Provision Of Complex Public Goods. (J. Bitzer & P. J. H. Schraeder, Eds.)*Social Science Research Network*, (July), 57–81. doi:10.2139/ssrn.588763

Bosch, J. (2009). From software product lines to software ecosystems. *SPLC '09 Proceedings of the 13th International Software Product Line Conference*, (Splc), 1–10.

Botzenhardt, A., Maedche, A., & Wiesner, J. Developing a domain ontology for software product management. , 2011 Fifth International Workshop on Software Product Management IWSPM 7–16 (2011). IEEE. doi:10.1109/IWSPM.2011.6046207

Boucharas, V., Jansen, S., & Brinkkemper, S. (2009). Formalizing Software Ecosystem Modeling, 41–50.

Brinkkemper, S, Ebert, C., & Versendaal, J. Proceedings of the First International Workshop on Software Product Management. , 2006 International Workshop on Software Product Management IWSPM06 RE06 Workshop 1–2 (2006). Ieee. doi:10.1109/IWSPM.2006.7

Brinkkemper, Sjaak. (1996). Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*, *38*(4), 275–280. doi:10.1016/0950-5849(95)01059-9

Capek, P. G., Frank, S. P., Gerdt, S., & Shields, D. (2005). A history of IBM's open-source involvement and strategy. *IBM Systems Journal*, *44*(2), 249–257. doi:10.1147/sj.442.0249

Crowston, K., Annabi, H., & Howison, J. (2003). Defining open source software project success. In Tbd (Ed.), *Information Systems Journal* (Vol. Paper 4, pp. 327–340). Citeseer. doi:10.1.1.10.6110

Crowston, K., & Howison, J. (2005). The social structure of free and open source software development. *First Monday*, *10*, 2–7. doi:10.5210/fm.v10i2.1207

Dhungana, D., Groher, I., Schludermann, E., & Biffl, S. (2010). Software ecosystems vs. natural ecosystems. *Ecosystems*, 96–102. doi:10.1145/1842752.1842777

German, D., & Mockus, A. (2003). Automating the measurement of open source projects. *ICSE'03 International Conference on Software Engineering* (pp. 63–68). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.136.3067

Gorschek T, K. H. B. (2011). International Software Product Management Association: Towards a Software Product Management certification. *2011 5th International Workshop on Software Product Management IWSPM 2011 Part of the 19th IEEE International Requirements Engineering Conference* (pp. 1–2). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-80555154897&partnerID=40&md5=a264a70aab6aad749b26da70e4628699

Harmsen, F., Brinkkemper, S., & Oei, J. L. H. (1994). Situational method engineering for informational system project approaches, 169–194. Retrieved from http://dl.acm.org/citation.cfm?id=647158.717382

Hauge, O., Osterlie, T., Sorensen, C.-F., & Gerea, M. (2009). An empirical study on selection of Open Source Software - Preliminary results. *2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development* (pp. 42–47). IEEE. doi:10.1109/FLOSS.2009.5071359

Hawkins, R. E. (2004). The economics of open source software for a competitive firm. *NETNOMICS: Economic Research and Electronic Networking*, *6*(2), 103–117. doi:10.1007/s11066-004-2717-z

Hevner, A., & Chatterjee, S. (2010). Design Science Research Frameworks. In A. Hevner & S. Chatterjee (Eds.), *Design Research in Information Systems* (Vol. 22, pp. 23–31). Springer US. doi:10.1007/978-1-4419-5653-8

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science Research in Information Systems. (A. Hevner & S. Chatterjee, Eds.)*Information Systems Journal*, *22*(1), 75–105. doi:10.1007/978-1-4419-5653-8

Immonen, A., & Palviainen, M. Trustworthiness Evaluation and Testing of Open Source Components. , Seventh International Conference on Quality Software QSIC 2007 316–321 (2007). doi:10.1109/QSIC.2007.4385514

Izquierdo-Cortazar, D., González-Barahona, J., Robles, G., Deprez, J., & Auvray, V. (2010). FLOSS Communities: Analyzing Evolvability and Robustness from an Industrial Perspective. (P. Agerfalk, C. Boldyreff, J. M. Gonzalez-Barahona, G. Madey, & J. Noll, Eds.)*Open Source Software New Horizons*, *319*, 336–341. doi:10.1007/978-3-642-13244-5_28

Jansen, S., Finkelstein, A., & Brinkkemper, S. (2009). A sense of community: A research agenda for software ecosystems. *31st International Conference on Software Engineering (ICSE 2009)*, 2–5.

Kabbedijk, J., & Jansen, S. (2011). Steering Insight: An Exploration of the Ruby Software Ecosystem. In B. Regnell, I. Weerd, & O. Troyer (Eds.), *Software Business* (Vol. 80, pp. 44–55). Springer Berlin Heidelberg. doi:10.1007/978-3-642-21544-5_5

Lungu, M. Towards reverse engineering software ecosystems. , 2008 IEEE International Conference on Software Maintenance 428–431 (2008). Ieee. doi:10.1109/ICSM.2008.4658096

Maki-Asiala, P., & Matinlassi, M. (2006). Quality Assurance of Open Source Components: Integrator Point of View. *30th Annual International Computer Software and Applications Conference (COMPSAC'06)* (pp. 189–194). IEEE. doi:10.1109/COMPSAC.2006.153

Manteli, C., Van De Weerd, I., & Brinkkemper, S. (2010). Bridging the gap between software product management and software project management. *ACM International Conference Proceeding Series*, 32–34. doi:10.1145/1961258.1961266

Michlmayr, M. (2005). Software Process Maturity and the Success of Free Software Projects. (K. Zielinski & T. Szmuc, Eds.)*Control*, *130*, 3–14. Retrieved from http://dl.acm.org/citation.cfm?id=1565145

Northrop, L. M., & Jones, L. (2004). Adopting Software Product Lines. *Software Product Lines*. Retrieved from http://www.springerlink.com/content/49x3yfqj1tutggg0

Okoli, C., & Schabram, K. (2010). A Guide to Conducting a Systematic Literature Review of Information Systems Research. *Sprouts Working Papers on Information Systems*, *10*(26), 1–49. Retrieved from http://sprouts.aisnet.org/867/1/OkoliSchabram2010SproutsLitReviewGuide.pdf

Parizi, R. M., & Ghani, A. A. A. (2010). Towards Automated Monitoring and Forecasting of Probabilistic Quality Properties in Open Source Software (OSS): A Striking Hybrid Approach. *2010 Eighth ACIS International Conference on Software Engineering Research, Management and Applications* (pp. 329–334). IEEE. doi:10.1109/SERA.2010.48

Raja, U., & Tretter, M. J. (2006). Investigating open source project success: A data mining approach to model formulation, validation and testing. *Neural Networks*, 1–7. Retrieved from http://www2.sas.com/proceedings/sugi31/071-31.pdf

Samoladas, I., Angelis, L., & Stamelos, I. (2010). Survival analysis on the duration of open source projects. *Information and Software Technology*, *52*(9), 902–922. doi:10.1016/j.infsof.2010.05.001

Samoladas, I., Gousios, G., Spinellis, D., & Stamelos, I. (2008). The SQO-OSS Quality Model: Measurement Based Open Source Software Evaluation. (B. Russo, E. Damiani, S. Hissam, B. Lundell, & G. Succi, Eds.)*Open Source Development Communities and Quality*, *275*, 237–248. doi:10.1007/978-0-387-09684-1

Schweik, C. M., & English, R. (2007). Identifying Success and Abandonment of Free/Libre and Open Source (FLOSS) Commons: A Preliminary Classification of Sourceforge.net projects. *Source*. ScholarWorks@UMass Amherst. Retrieved from http://scholarworks.umass.edu/cgi/viewcontent.cgi?article=1000&context=opensource

Sethanandha, B. D., Massey, B., & Jones, W. Managing open source contributions for software project sustainability. , Technology Management for Global Economic Growth PICMET 2010 Proceedings of PICMET 10 1–9 (2010). IEEE. Retrieved from http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5602062

Stewart, K J, & Ammeter, T. (2002). An exploratory study of factors influencing the level of vitality and popularity of open source projects. *Proceedings of the TwentyThird International Conference on Information Systems* (Vol. ICIS 2002, pp. 1–5). Retrieved from http://aisel.aisnet.org/icis2002/88

Stewart, K., Ammeter, A., & Maruping, L. A Preliminary Analysis of the Influences of Licensing and Organizational Sponsorship on Success in Open Source Projects. , 7 Proceedings of the 38th Annual Hawaii International Conference on System Sciences 197c–197c (2005). Ieee. doi:10.1109/HICSS.2005.38

Stewart, Katherine J, Ammeter, A. P., & Maruping, L. M. (2006). Impacts of License Choice and Organizational Sponsorship on User Interest and Development Activity in Open Source Software Projects. *Information Systems Research*, *17*(2), 126–144. doi:10.1287/isre.1060.0082

Subramaniam, C., Sen, R., & Nelson, M. L. (2009). Determinants of open source software project success: A longitudinal study. *Decision Support Systems*, *46*(2), 576–585. doi:10.1016/j.dss.2008.10.005

The Linux Foundation Releases Annual Linux Development Report. (2012).*Linux Kernel Development*. Retrieved March 19, 2013, from http://www.linuxfoundation.org/news-media/announcements/2012/04/linux-foundation-releases-annual-linux-development-report

Torres, M. R. M., Toral, S. L., Perales, M., & Barrero, F. Analysis of the Core Team Role in Open Source Communities. , 2011 International Conference on Complex Intelligent and Software Intensive Systems 109–114 (2011). IEEE. doi:10.1109/CISIS.2011.25

Välimäki, M. (2003). Dual Licensing in Open Source Software Industry. *Systemes d´Information et Management*, *8*(1), 63–75. Retrieved from http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1261644

Van De Weerd, I., & Brinkkemper, S. (2008). Meta-modeling for situational analysis and design methods. (M. R. Syed & S. N. Syed, Eds.)*Handbook of Research on Modern Systems Analysis and Design Technologies and Applications*, 35. doi:10.4018/978-1-59904-887-1

Van De Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., & Bijlsma, L. (2006). Towards a Reference Framework for Software Product Management. *14th IEEE International Requirements Engineering Conference RE06*, *0*(014), 319–322. doi:10.1109/RE.2006.66

Ververs, E., Bommel, R. Van, & Jansen, S. (2011). Influences on developer participation in the Debian software ecosystem. *MEDES ’11 Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, 89–93.

Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011). The agile requirements refinery: Applying SCRUM principles to software product management. *Information and Software Technology*, *53*(1), 58–70. doi:10.1016/j.infsof.2010.08.004

Von Hippel, E. (2001). Learning from open-source software. *MIT Sloan management review*, *42*(4), 82–86.

Wahyudin, D., Mustofa, K., Schatten, A., Biffl, S., & Tjoa, A. M. (2007). Monitoring the "health" status of open source web-engineering projects. *International Journal of Web Information Systems - IJWIS*, *3*(1/2), 116–139. doi:10.1108/17440080710829252

Wahyudin, D., Schatten, A., Mustofa, K., Biffl, S., & Tjoa, A. M. (2006). Introducing"Health"Perspective in Open Source Web-Engineering Software Projects, Based on Project Data Analysis. *International Conference on Information Integration and Web-based Applications & Services (IIWAS)* (pp. 269–278). Retrieved from http://www.isis.tuwien.ac.at/node/13054

Weerd, I. Van De. (2008). Meta-modeling for situational analysis and design methods. *systems analysis and design*.

Weiss, D. (2005). Measuring success of open source projects using web search engines. (M. Scotto & G. Succi, Eds.)*Technology*, (December), 93–99 TS – BeitraginSammelband. Retrieved from http://eprints.lincoln.ac.uk/76/

West, J., & O'mahony, S. (2008). The Role of Participation Architecture in Growing Sponsored Open Source Communities. *Industry Innovation*, *15*(2), 145–168. doi:10.1080/13662710801970142

Wu, C.-G., Gerlach, J. H., & Young, C. E. (2007). An empirical analysis of open source software developers' motivations and continuance intentions. *Information & Management*, *44*(3), 253–262. doi:10.1016/j.im.2006.12.006

Yu, E., & Deng, S. (2011). Understanding Software Ecosystems : A Strategic Modeling Approach. *Proceedings of the Workshop on Software Ecosystems* (pp. 65–76).

Yu, L., Ramaswamy, S., Lenin, R. B., & Narasimhan, V. L. (2009). Time series analysis of open-source software projects. *Proceedings of the 47th Annual Southeast Regional Conference on - ACM-SE 47* (p. 1). New York, New York, USA: ACM Press. doi:10.1145/1566445.1566531

Zhou, Y., & Davis, J. (2005). Open source software reliability model. *Proceedings of the fifth workshop on Open source software engineering - 5-WOSSE* (Vol. 30, pp. 1–6). New York, New York, USA: ACM Press. doi:10.1145/1083258.1083273

10. Appendix

APPENDIX A: SYSTEMATIC LITERATURE REVIEW

COMBINATION ID	WORD COMBINATION
1	open source health
2	open source vitality
3	open source activity
4	open source status
5	open source analysis
6	open source growth
7	open source community health
8	open source community vitality
9	open source community activity
10	open source community status
11	open source community analysis
12	open source community growth
13	open source project health
14	open source project vitality
15	open source project activity
16	open source project status
17	open source project analysis
18	open source project growth
19	open source component health
20	open source component vitality
21	open source component activity
22	open source component status
23	open source component analysis
24	open source component growth
25	ecosystem health
26	ecosystem vitality
27	ecosystem activity
28	ecosystem status
29	ecosystem analysis
30	ecosystem growth
31	software ecosystem health
32	software ecosystem vitality
33	software ecosystem activity
34	software ecosystem status
35	software ecosystem analysis
36	software ecosystem growth

Table 10 - Systematic Literature Review: Keyword combinations

KEY WORD ID	ORIGIN: GOOGLE SCHOLAR	ORIGIN: MICROSOFT ACADEMIC	ACCEPTABLE BY ABSTACT	ACEPTABLE BY CONTENT	TITLE
15		yes	no		A stage model of open source activities: An exploratory analysis on open source repository
11	yes		yes	no	Analysis and modeling of the open source software community
8	yes		yes	yes	Defining open source software project success
18	yes		yes	yes	Investigating open source project success: a data mining approach to model formulation, validation and testing
1	yes		no		Web GIS in practice IV: publishing your health maps and connecting to remote WMS sources using the Open Source UMN MapServer and DM Solutions
1	yes		no		Why Open Source Software/Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers!
33	yes		yes	yes	A framework for analysing and visualising open source software ecosystems
35		yes	no		A method for analyzing software product line ecosystems
10		yes	no		A network perspective on open source software development: Team formation and community participation
2		yes	yes	yes	A Preliminary Analysis of the Influences of Licensing and Organizational Sponsorship on Success in Open Source Projects
32		yes	no		A state of the art review on software project performance management
21		yes	yes	no	A Survey on Firms' Participation in Open Source Community Projects
6		yes	no		A Systematic Review of Studies of Open Source Software Evolution
33	yes		no		Activity theory for OSS ecosystems
24		yes	no		Adoption of open source software: Is it the matter of quality ?
1	yes		yes	yes	An empirical analysis of open source software developers' motivations and continuance intentions
2		yes	yes		An Empirical Investigation of Defect Management in Free/Open Source Software Projects
2		yes	yes	yes	An Empirical Study on Selection of Open Source Software - Preliminary Results
15		yes	no		An Empirical Study on the Relationship Between Software Design Quality, Development Effort and Governance in Open Source Projects
3		yes	no		An Exploratory Long-Term Open Source Activity Analysis: Implications from Empirical Findings on Activity Statistics
2	yes		yes	yes	An exploratory study of factors influencing the level of vitality and popularity of open source projects
2	yes		no		An Open Source Approach to Developing Software in a Small Organization
11		yes	yes	no	Analysis and Modeling of Open Source Software Community

3		yes	yes	no	Analysis of Activity in the Open Source Software Development Community
36		yes	no		Analysis of projects and volunteer participation in large scale free and open source software ecosystem
11		yes	yes	yes	Analysis of the Core Team Role in Open Source Communities
23		yes	yes	no	Applying Service-Oriented Software Measurement to Derive Quality Indicators of Open Source Components
7		yes	yes	yes	Assessing the Health of Open Source Communities
2		yes	yes	yes	Automating the Measurement of Open Source Projects
9		yes	no		Case study of company's relationship with open source community in open source software development
3	yes		yes	no	Cave or community?: An empirical examination of 100 mature open source projects
21		yes	no		Challenges of the Open Source Component Marketplace in the Industry
2		yes	yes	no	Characteristics of Open Source Projects
12		yes	no		Communication Networks in an Open Source Software Project
8		yes	no		Community effort in online groups: Who does the work and why?
12	yes		no		Community-based production of open-source software: What do we know about the developers who participate?
1	yes		no		Comparing motivations of individual programmers and firms to take part in the open source movement: From community to business
5		yes	no		Contractual Relationships in Open Source Structures
17		yes	no		Contrasting Community Building in Sponsored and Community Founded Open Source
7		yes	yes	no	Coordination Dynamics in Free/Libre Open Source Software Development
1		yes	no		Critical issues associated with adoption and use of open source software in public sector: insights from Tanzania
1		yes	no		Customization of Open Source Software in Companies
17		yes	yes	no	Detecting Agility of Open Source Projects Through Developer Engagement
15	yes		yes	yes	Determinants of open source software project success: A longitudinal study
1	yes		no		Developing an information systems infrastructure with open source software
1	yes		no		Entry strategies under competing standards: Hybrid business models in the open source software industry
23		yes	yes		Evaluating Quality of Open Source Components for Reuse-Intensive Commercial Solutions
5	yes		no		Evolution in open source software: A case study
9		yes	no		Evolution of Open Source Communities
6		yes	no		Evolution patterns of open-source software systems and communities
22		yes	yes	yes	FLOSS Communities: Analyzing Evolvability and Robustness from an Industrial Perspective
6		yes	no		Growth, evolution, and structural change in open source

					software
9	yes		no		How open is open enough?: Melding proprietary and open source platform strategies
1		yes	no		How Open Source Can Still Save the World
14		yes	yes	no	Identifying knowledge brokers that yield software engineering knowledge in OSS projects
14		yes	yes	yes	Identifying Success and Abandonment of FLOSS Commons: A Classification of Sourceforge.net Projects
8	yes		yes	yes	Impacts of license choice and organizational sponsorship on user interest and development activity in open source software projects
31	yes		no		Information Technology Ecosystem: Structure, Health, and Performance
13		yes	yes	yes	Introducing Health Perspective In Open Source Web-Engineering Software Projects, Based On Project Data Analysis
2		yes	no		Issues of dependability in open source software development
1	yes		no		Legal implications of open-source software
15		yes	yes	yes	Managing open source contributions for software project sustainability
20	yes		no		Measuring open source software success
15		yes	yes	yes	Measuring Success of Open Source Projects Using Web Search Engines
12		yes	no		Measuring the evolution of open source software systems with their communities
3	yes		yes	no	Mission-critical development with open source software: Lessons learned
8		yes	no		Monetary donations to an open source software platform
4	yes		yes	yes	Monitoring the “health” status of open source web-engineering projects
10	yes		yes	no	Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel
3	yes		yes	no	Motivation, governance, and the viability of hybrid forms in open source software development
1		yes	no		O3-DPACS system: challenges and original solutions in developing an open source project for the PACS critical system
14	yes		no		Open borders? immigration in open source projects
1		yes	no		Open source development: a hybrid in innovation and management theory
2	yes		no		Open source intelligence
17		yes	yes	yes	Open Source Project Categorization Based on Growth Rate Analysis and Portfolio Planning
1	yes		no		Open source software development as a special type of academic research
4		yes	no		Open Source Software Development Projects: Determinants of Project Popularity
2	yes		no		Open source software development should strive for even greater code maintainability

18		yes	yes	yes	Open source software reliability model: an empirical approach
7	yes		no		Open source software user communities: A study of participation in Linux user groups
3		yes	no		Open strategies, open source strategies and the issue of value capture in the software industry
21		yes	yes	yes	Quality Assurance of Open Source Components: Integrator Point of View
10		yes	no		Quantitative Study of Open Source Immigration
6		yes	yes	no	Relationships between open source software companies and communities: Observations from Nordic firms
11		yes	no		Reusing Open-Source Software and Practices: The Impact of Open-Source on Commercial Vendors
2		yes	no		Reverse Engineering the Bazaar: Collaboration and Communication in Open Source Development
34		yes	no		Revisiting the concept of components in software engineering from a software ecosystem perspective
16		yes	yes	no	Sampling Open Source Projects from Portals: Some Preliminary Investigations
35		yes	no		SECONDA: Software Ecosystem Analysis Dashboard
11	yes		no		Social status in an open-source community
14		yes	yes	yes	Software Process Maturity and the Success of Free Software Projects
36		yes	yes	yes	Software Quality Assessment of Open Source Software
1	yes		no		Striking a balance between trust and control in a virtual organization: a content analysis of open source software case studies
17		yes	yes	yes	Survival analysis on the duration of open source projects
8		yes	yes	no	The adoption of open source software in business models: a Red Hat and IBM case study
14	yes		yes	no	The ecology of open-source software development
12	yes		no		The governance of open source initiatives: what does it mean to be community managed?
6		yes	no		The impact of ideology on effectiveness in open source software development teams
2	yes		no		The many meanings of open source
11		yes	no		The Open Source Software Community Structure
1		yes	yes	no	The Practice of Free and Open Source Software Processes
12		yes	yes	yes	The Role of Participation Architecture in Growing Sponsored Open Source Communities
35		yes	no		The Small Project Observatory: Visualizing Software Ecosystems
1		yes	yes	yes	The SQO-OSS Quality Model: Measurement Based Open Source Software Evaluation
1	yes		no		The success of open source
3		yes	yes	no	The Total Growth of Open Source
15		yes	yes	yes	Time series analysis of open-source software projects
16	yes		yes	yes	Towards a portfolio of FLOSS project success measures
14		yes	yes	yes	Towards Automated Monitoring and Forecasting of Probabilistic Quality Properties in Open Source Software

					(OSS): A Striking Hybrid Approach
16		yes	no		Trust in Virtual Communities involved in Free/Open Source Projects: An Empirical Study
23		yes	yes	yes	Trustworthiness Evaluation and Testing of Open Source Components
3	yes		yes	no	Two case studies of open source software development: Apache and Mozilla
4		yes	no		Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects
1		yes	no		Understanding the Open-Source Software Development Process: a Case Study with CVSChecker
2		yes	no		Up from alchemy [open source development]
3		yes	no		Value-Creating Networks Approach to Open Source Software Business Models
31		yes	no		What is a healthy ecosystem?
12	yes		no		When does community participation enhance the performance of open source software companies?
4		yes	no		Why Do Developers Contribute to Open Source Projects? First Evidence of Economic Incentives
3	yes		no		Why hackers do what they do: Understanding motivation and effort in free/open source software projects
15	yes		no		Working for free? Motivations of participating in open source projects

Table 11 - Systematic Literature Review: Candidate articles and acceptance status

APPENDIX B: INTERVIEW PROTOCOLS

ASSESSMENT OF CURRENT PRACTICES AND OSC VITALITY INDICATORS AT IBM

Participation Invite

*"Dear ***,*

I am a MSc student in Business Informatics, currently conducting research for my master's thesis on vitality analysis of open source communities / components utilized in commercial software products. A search on Bluepages revealed you as an expert in this domain and I was hoping to be able to schedule a brief phone interview with you, at your convenience."

Introduction

During the interview, all interviewees received the same, common introduction:

"The purpose of my research is to design a method for software product managers that will enable them to systematically asses the vitality of open source communities that their managed software product depends on. I am specifically looking for open source components in commercial applications, as open source plays an increasingly important role in proprietary software development and integrating OSCs into commercial applications has a number of risks that are presently difficult to assess. The purpose of this interview is to gain an understanding of OSS practices at IBM, explore your view on OSC / OSC vitality indicators and gain insights into practical experiences of OSS implementation in commercial software products."

Current Open Source related practices of the interviewee

All interviewees were asked questions regarding their understanding of IBM open source activates and their respective contribution or practice of them. As internal documentation on OSC utilization exists, a focus was given to questions looking at potentially undocumented practices and OSC related experiences of the interviewees.

1. What are your experience with IBM's Open Source Core Team (OSCT) and the OSSC process?
2. How do you use open source in you role's activities?
3. Next to the OSCT documentation, are there processes or systematic considerations for the use of open source in IBM's software portfolio or individual projects?
4. Do you, in your role as ****, actively utilize open source software or interact with open source communities?

Open Source Vitality

Interviewees were asked to describe their experiences with open source health and indicate how they would assess the quality and vitality of an open source component:

1. How would you define a vital OS community?
2. How and which aspects of OSCs would you measure or evaluate to assess the communities health?
3. Have you ever evaluated OS components or the underlying communities? If yes, what was your approach and which metrics have you used?

Open Source role in SPM Activities

As previously mentioned, IBM's hierarchy does not incorporate the role of a software product manager or a direct equivalent to it. Thus, the following questions are aimed at the interviewee's general responsibilities and activities in regard to open source.

1. Which role das Open Source play in your activities in software product management?
2. Do you, in your role as ****, give strategic considerations to the use Open Source?
3. Do you manage or have strategic relationships with Open Source communities?

OSC Health Analysis Method Validation

Introduction

During the phone interview, all interviewees received the same, common introduction:

"The purpose of my research is to design a method for software product managers that will enable them to systematically asses the vitality of open source communities that their managed software product depends on. I am specifically looking for open source components in commercial applications, as open source plays an increasingly important role in proprietary software development and integrating OSCs into commercial applications has a number of risks that are presently difficult to assess.

During this interview I would like to validate a method that was designed to answer my research purpose. I am particularly interested in hearing your opinion on its applicability when assessing open course communities. Please feel free to write any observations and impressions you have immediately on the provided pages"

General Method Questions

Following the domain introduction, the interviewees were asked general questions regarding the presented method fragments. The intention of this interview section was to establish an understanding of the clarity and applicability of the method from a usability standpoint.

1. Is the purpose of situational method design clear or did you have difficulties following the introduction?
2. Did you have any issues understanding the selection rationale and can you see yourself applying such a heuristic for fragment selection?
3. Do you find the selection rationale suitable for fragment selection?
4. Did you understand the approach for method assembly and would you apply such a technique in your OSC related activities?
5. Did you understand the approach for method calibration and would you apply such a technique in your OSC related activities?

Selection Scenarios

Each interviewee was presented with the below scenarios and was given a print out of all introduced method fragments. After having read the scenarios, they were asked to assemble a situational method based on the presented requirements. Furthermore, the interviewees were encouraged to write, annotate and highlight the printed fragments and add any remarks and observations they made as they were assembling the fragments.

1. **Small IT service Provider**

A regional IT provider with 10 employees would like to use an open source based automation tool for a customer's built processes. A discovered OSC that appears to have suitable functionality is approximately two years old and relies on two core developers that carry on a majority of the coding activities. Due to the specific nature of the tool, the community is relatively small and mostly consists of software developers who occasionally contribute patches.

Please create a suitable method to assess the vitality of this OSC and decide whether you can deliver a solution to the customer, who intends to base its internal development processes on the tool for the next 3 years.

2. **SME Software Developer with one core software product**

You are responsible for managing a highly customizable decision support system designed for the energy and utilities industry. In order to focus on the statistical aspect of the product, you decide to use an OSC JavaScript framework for data visualization. The candidate OSC is about four years old and has already produced a number of popular releases, however new projects with similar functionality are frequently appearing on Github and you are not certain if core developers will stay with the existing project or choose to invest their efforts in one of the new OSCs.

Please design a situational method that you find suitable to assess the current community and describe your approach in dealing with the current uncertainties regarding the OSC.

3. **Large Standard and Custom Software Developer working on a variety of different projects**

In your role as a software product manager for a business process suite, you frequently validate the suitability of OSCs for integration purposes. As your managed application has a lifecycle of minimum five years, you must ensure that every component can be supported for an extensive period of time. You are evaluating an open source, Java based MVC framework and would like to increase your understanding of the community's vitality and long term sustainability. The OSC in questions has existed for three years, has an active member base and mature internal processes.

Please design a situational method that can be used to assess the long term survivability of the community in question.

Scenario feedback questions

After completion of the initial question round and the scenario driven method assembly, all interviewees were asked to respond to a set of questions regarding the experiences they gained with situational method assembly for OSC vitality analysis.

All interviewees were asked the following questions:

1. Did you find the provided situational scenarios sufficient in order to choose suitable method fragments?
2. Were the three selection rationale items clear and could you match them to the presented cases?
3. How did you perceive the process of assembling the fragments?
4. Were the individual fragment actions and deliverables easy to understand or did you have trouble with certain aspects of the fragments' structure?
5. Were the fragment categories and their deliverables clear?
6. Do you believe that the introduced deliverables are suitable for vitality assessment?
7. How do you see the process of community data gathering in light of the presented methods?
8. Do you see any obstacles to data gathering and data evaluation?
9. Do you see any obstacles to fragment calibration?
10. Would you use a structured approach, such as the presented method in your OSC related activities? If yes, please elaborate on how you would apply it. If no, please explain why you find it unsuitable.

APPENDIX C: ANONYMISED INTERVIEWEE LIST – IBM BENELUX

Following the introduction of the interview protocol in appendix B, Table 12 depicts a list of OSC experts, interviewed at IBM Benelux. For privacy purposes the interviewee names have been replaced with identification numbers. The full names are known to the author.

Interviewee ID	Interviewee Job Title
I-1	Open Source And Middleware Developer
I-2	IT Specialist Data Integration
I-3	IT specialist with a preference for testing and test automation
I-4	Linux deployment specialist / Server Support Specialist
I-5	Senior IT Consultant / Systems Management Specialist
I-6	SO Delivery, Server Systems Operations
I-7	Rational Account Manager and IT Specialist, IBM Certified
I-8	Program Manager, EM Domain and IDEs
I-9	Client Technical Architect
I-10	Senior Cloud Solutions Architect

Table 12 - Anonymised Interviewee List

APPENDIX D: COMPLETE OSC HEALTH ANALYSIS METHOD

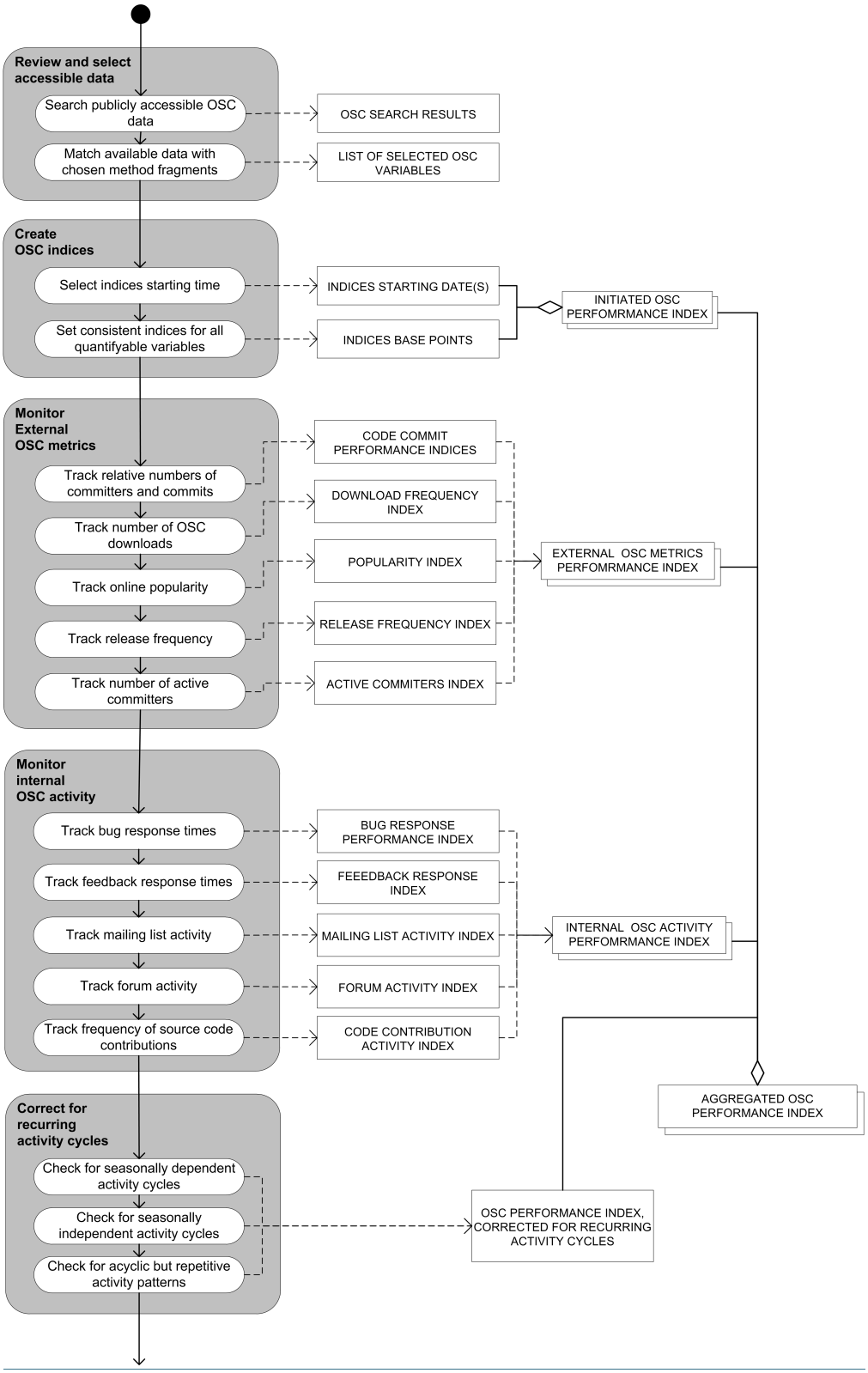

Figure 28 – Complete OSC Health Analysis Method: Part 1

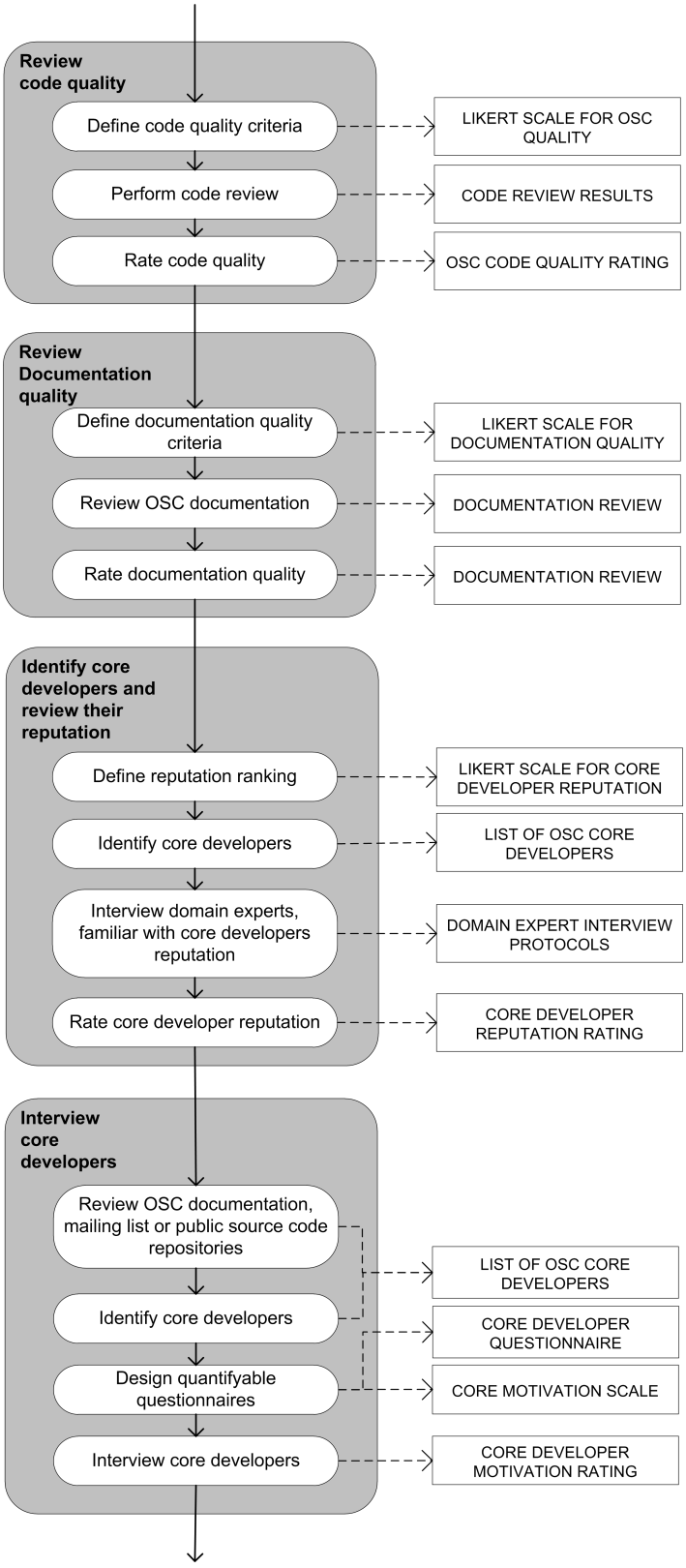

Figure 29 – Complete OSC Health Analysis Method: Part 2

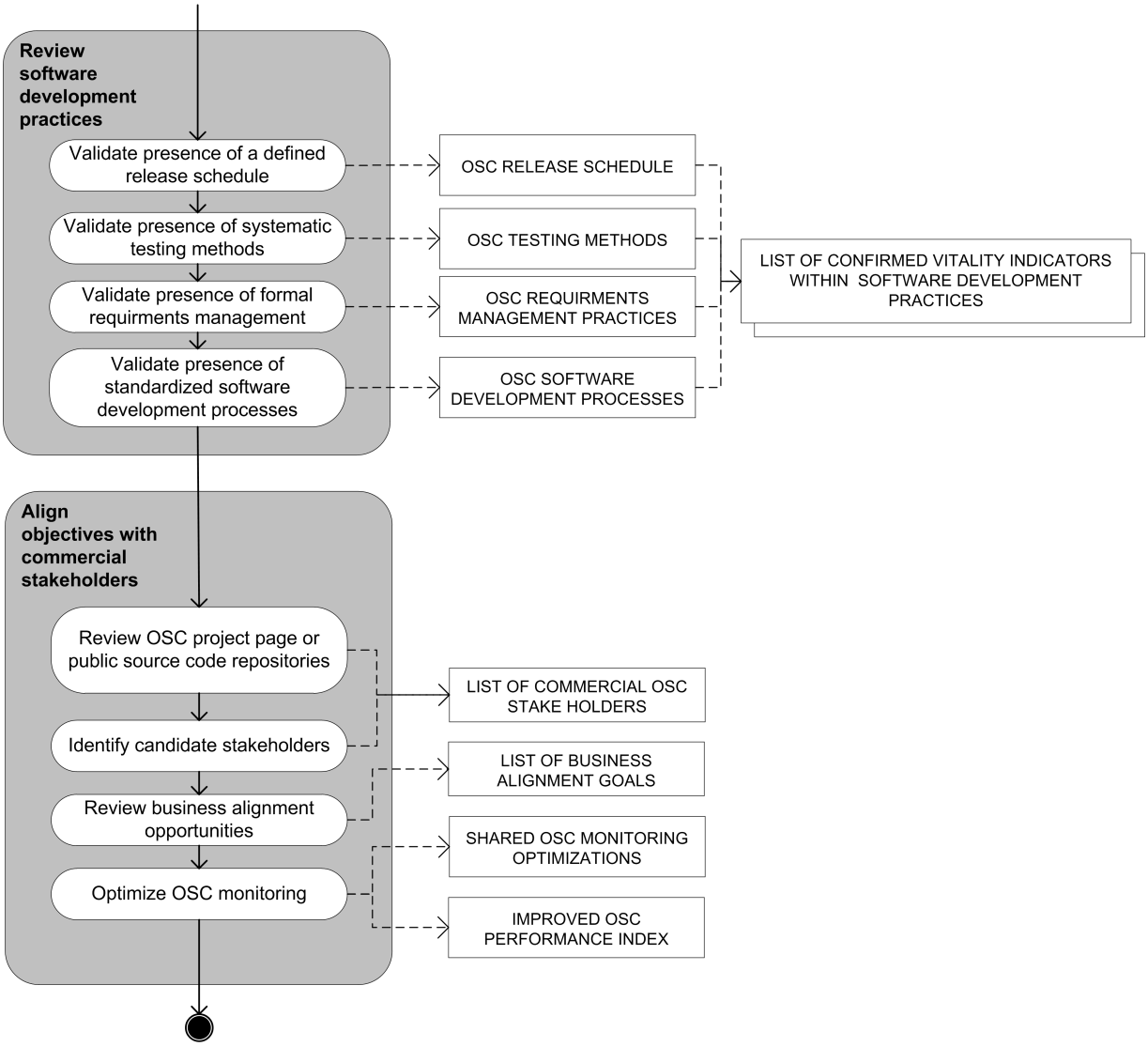

Figure 30 – Complete OSC Health Analysis Method: Part 3

APPENDIX E: EXAMPLE GRAPHS FOR OSC HEALTH METHOD APPLICABILITY

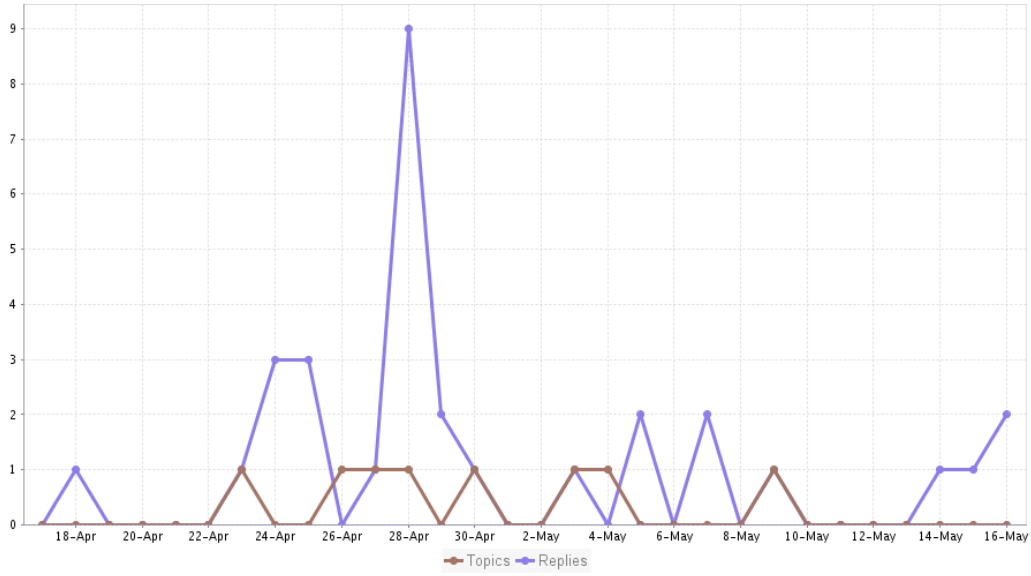

Figure 31 – Sample OSC: Forum activity in number of Posts for jQuery Core

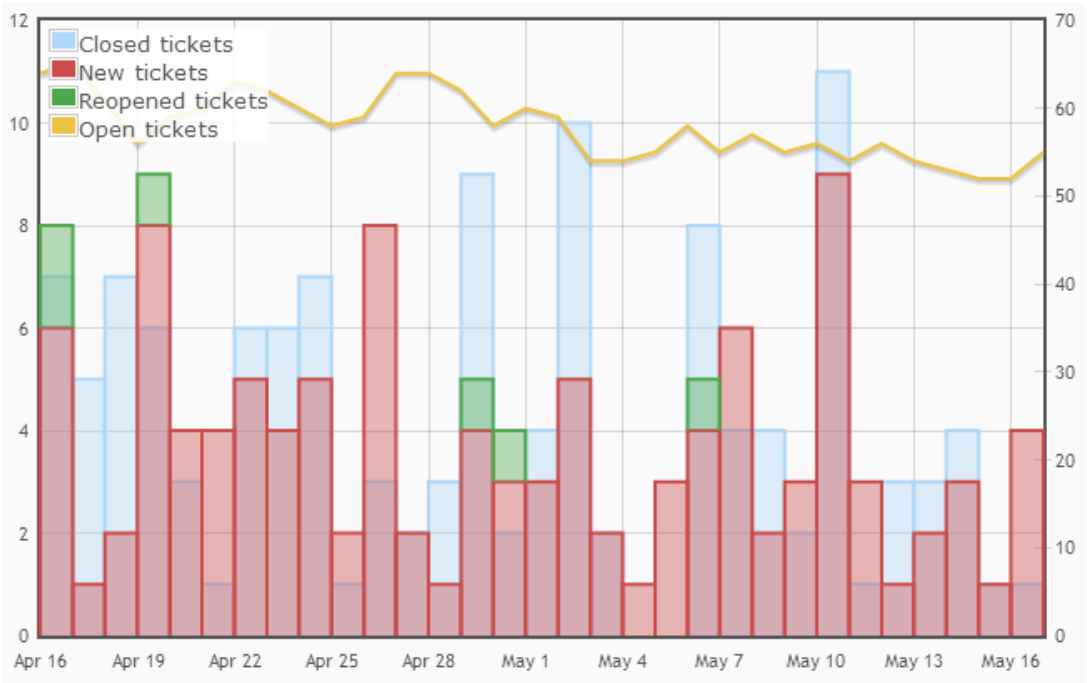

Figure 32 - Sample OSC: Community Issue tracker for jQuery Core