Avoin lähdekoodi käytössä

Näkökulmia avoimen lähdekoodin ohjelmistojen käyttöön ja kehittämiseen julkisella ja yksityisellä sektorilla

Teemu Mikkonen & Tere Vadén (toim.)
teemu.mikkonen@uta.fi, tere.vaden@uta.fi

Kannen suunnitellut Sanna Säynäjäkangas
Julkaisusarjan taiton suunnitellut Simo Kaupinmäki

ISBN 978-951-44-7423-1
Hypermedialaboratorion tutkimusraportteja, 7.
Mikkonen, Teemu

Tampereella heinäkuussa 2008
Tampereen yliopiston hypermedialaboratorio
http://www.uta.fi/hyper/

Sisällys

1. Alkusanat

Vapaan ja avoimen lähdekoodin ohjelmistot (VALOt; Free and Open Source Software, FOSS) ovat vakiinnuttaneet paikkansa paitsi ohjelmistojen kehittäjien ja heidän yhteisöjensä myös ohjelmistoteollisuuden ja tavallisten käyttäjien keskuudessa. Ilmiö on muuttumassa: puhutaan esimerkiksi Open Source Software (OSS) 2.0-vaiheesta, jossa OS-ohjelmistojen käyttö ja kehittäminen on isojen ohjelmistoyritysten arkipäivää. Samalla lisääntyy myös VALOjen tutkimus Suomessa ja maailmalla. Kirjoja ja artikkeleita ilmestyy kiihtyvällä tahdilla ja kansainvälinen konferenssitoiminta on vilkasta. Tutkimustietoa tarvitaan niin ohjelmistojen laadusta, kehittäjäyhteisöistä, liiketoimintamahdollisuuksista, käyttökokemuksista kuin lainsäädännön muutoksistakin.

Tampereen yliopiston Hypermedialaboratorio ja Suomen Open Source -keskus (COSS) järjestivät Tampereella 1.10. 2007 järjestyksessään toisen VALO-tutkijoiden tapaamisen. Tapaamisten tarkoituksena on verkottaa alan suomalaisia tutkijoita ja saattaa uusimmat tutkimushankkeet kaikkien tietoon. Tapaamisessa kuultiin esitykset kahdeksasta käynnissä olevasta tutkimushankkeesta ja lisäksi tunnetun Open Source asiantuntijan Stephen Wallin yhteenveto kansainvälisen FOSS -tutkimuksen tilasta.

Tässä tutkimusraportissa julkaistaan neljä tutkijatapaamisen innoittamaa artikkelia. Ensimmäinen niistä koskee OS-ohjelmistohankkeiden arviointia. Kirjoittajat Timo Aaltonen, Nina Helander ja Marko Seppänen Tampereen teknilliseltä yliopistolta käyvät artikkelissa läpi neljä erilaista arviointimenetelmää, ja tulostensa perusteella hahmottelevat parannettua mallia, joka ottaa paremmin huomioon ohjelmiston käyttötarkoituksen ja käyttäjän halukkuuden osallistua ohjelmiston kehittämiseen.

Toinen artikkeli, Outi Grotenfeltin ”Open Sourcen käyttö julkisella sektorilla Suomessa”, tarkastelee tapaustutkimusten kautta OS-ohjelmistojen käyttöönoton motivointia, käyttäjien tyytyväisyyttä ja ohjelmistojen käytettävyyttä suomalaisissa julkisen sektorin instituuteissa. Tutkimuksen tulokset ovat mielenkiintoisia. Erityisesti ohjelmistotuki – joka on helpompi järjestää, kun IT-toimintoja johdetaan keskitetysti – ja johdon sitoutuminen uusiin ohjelmistoihin tuntuvat olevan tärkeitä OS-softaan siirtymisen edistäjiä. Vastaavasti käyttäjien iällä, sukupuolella tai IT-taustalla ei tunnu olevan kovin suurta vaikutusta OS-ohjelmistojen siirtymän onnistumisen kannalta.

Kolmannessa artikkelissa Juho Lindman arvioi OS-tutkimuskentän muutoksia. OS-ohjelmistot eivät enää vaadi erityistä perustelua; on siirrytty normalisoituneeseen tilanteeseen. Samalla yritykset ovat yhä enemmän mukana OS-kehityksessä ja käyttävät OS-toimintatapoja osana omaa toimintaansa. Lindmanin mukaan tämä vaatii muutosta myös tutkijoiden asenteessa, kun kehityksen avoimuus ei enää ole itseisarvo tai keskeinen ideologinen tekijä. Hän perää huomiota OS-ilmiöiden monimuotoisuudelle ja uusien ilmiöiden, kuten avoimen ja suljetun koodin hybridien, syntymiselle.

Neljäs artikkeli koskee OS-kehittäjäyhteisöjen sosiologiaa. Teemu Mikkonen käsittelee kehittäjäyhteisöjen perustoja ranskalaisen klassikon Emile Durkheimin käsitteiden avulla. Mielenkiintoisella tavalla Mikkonen pohtii, onko Durkheimin esittämien yhteisöllisen ja egoistisen solidaarisuuden lisäksi tunnistettavissa erityinen tiedollisen solidaarisuuden muoto, joka luonnehtii juuri VALO-yhteisöjä ja muita samaa metodologiaa noudattavia hankkeita, kuten Wikipediaa.

Tere Vadén

2. Towards Successful Open Source Project Evaluation

Timo Aaltonen, Nina Helander & Marko Seppänen
Tampere University of Technology

Abstract

This paper discusses open source project evaluation. We start the paper by explaining why there is a need for a comprehensive evaluation and continue by going through four existing evaluation models. On the basis of comparison of these models, we select one of the models and assess it by evaluating one specific open source project, Gnome project. In the light of our case, we discuss the shortcomings of the existing evaluation models and identify the areas, which especially would need more attention in the evaluation. We conclude the paper by drafting an improved evaluation model, which offers a more comprehensive view to open source software evaluation than the previous ones.

Keywords: open source software, evaluation, model, business

2.1. Introduction

Companies are increasingly encountering selection situations in which they have several different possibilities to choose from when bundling software components. The possibilities include, at least, in-house development, acquiring proprietary software, acquiring or partnering with another firm with needed capabilities, and selecting an OSS alternative from the pool available. Some of these processes are better understood, but the selection and evaluation of OSS alternatives still provides challenges to companies. As in many cases software choice implies commitment to a community, with the risks and potentials of the commitment often lasting for several years, the decision to choose a specific OSS project should not be done by light arguments. Instead, proper evaluation of the alternatives should be carried out.

Basically, the evaluation could be done as simply as taking two brilliant software engineers and by giving them two days time to test the software and analyze its applicability. This evaluation approach is at best very time and cost efficient and if the two engineers really know what they are doing, the evaluation can be very high-quality. However, this evaluation approach has also weaknesses. It is a great risk that the two engineers don't see the whole picture, but for example forget to take a closer look to such important issues as license types, marketing, sales, healthiness of the community etc. Our argument is that to successfully take all these important aspects into account, there is a need for a systematic process for evaluation open source software and projects.

In the following section, we will present four existing OSS evaluation models and analyze their strengths and potential weaknesses. This paper is extracted and slightly modified from the OSSI project's report (Helander et al. 2007).

2.2. Different evaluation tools for OSS projects

In the following sub-sections, we provide descriptions of four evaluation tools which are developed for assessing open source products or projects, namely Optaros' Enterprise Readiness (ER) model, Open Source Maturity Model (OSMM) by B. Golden, a model for Qualification and Selection of Open Source Software (QSOS), and finally, Business Readiness Rating (BRR). These models can be regarded as the "best-of-breed" according to our knowledge. The following presentations of evaluation models are primarily based on the core sources of each model including e.g. the model's website etc.

Model #1: OPTAROS' Model

Optaros (2007) is an international consulting and systems integration firm that has created a catalog to provide a list of products best suited for today's enterprises. The catalog and its on-line version complemented with case studies and other information are available in http://www.eosdirectory.com. Only the products that match the enterprise benchmark in terms of functionality, community backing as well as maturity are listed. Technologies/projects are evaluated against four criteria:

Functionality is compared with what is usually needed (e.g. in commercial products).

Community demonstrates activity and support of the community behind the project.
Maturity measures quality and robustness of a software product
Trend indicates the expected future progress of the software product

Enterprise Readiness (ER) rating indicates how capable an open source software is to cope with the needs and requirements of midsize and large enterprises and organizations. ER-rating is indicated by one, two or three stars (Optaros' catalog does not list products that do not at least meet the one star level).

Table 1. Optaros ER ratings for Gnome, Debian, MySQL and Eclipse.

Product	Version	Description/ URL	License	Support	Function-ality	Comm-unity	Maturity	ER-Rating	Trend
Gnome	2.14	Graphical desktop environment for Linux http://www.gnome.org/	GPL	Community	✓ ✓ ✓ ✓	✶ ✶ ✶ ✶	✷ ✷ ✷ ✷	◆ ◆ ◆	→
Debian GNU /Linux	3.1	Widely used Linux distribution http://www.debian.org	GPL	Community	✓ ✓ ✓ ✓	✶ ✶ ✶ ✶	✷ ✷ ✷ ✷	◆ ◆	→
MySQL	5.0.22	Widely used open source relational database http://www.mysql.com/	GPL	Prof / Community	✓ ✓ ✓ ✓	✶ ✶ ✶ ✶	✷ ✷ ✷ ✷	◆ ◆ ◆	↗
Eclipse	3.2	Leading Java IDE. The foundation was inherited from IBM VisualAge http://www.eclipse.org/	Eclipse Public License	Community	✓ ✓ ✓ ✓	✶ ✶ ✶ ✶	✷ ✷ ✷ ✷	◆ ◆ ◆	↗

According to Optaros, for many applications, an open source product with a smaller functionality scope might be the better choice than a more complex one that does more than what is needed. Moreover, in other situations, a simpler tool may be easier to integrate than a comprehensive one using another technology.

Model #2: Open Source Maturity Model (OSMM)

The Open Source Maturity Model™ (OSMM) was created by Bernard Golden in 2004. The OSMM provides a framework to determine maturity level of an open source product. Its purpose is to enable a quick assessment of the maturity level of a given open source product. It offers great power to organizations evaluating the production readiness of an open source product, and to demonstrate its power a real assessment for JBOSS is performed.

The basic question is how to choose the best candidate? By using the OSMM, the products can be ranked according to their OSMM scores. Golden states that the model is designed to enable one or two people to spend no more than three to five days developing an overall maturity score for a product (i.e., to carry out a desk check).

The OSMM assesses a product's maturity in three phases:
Assess each product element's maturity and assign a maturity score
Define a weighting for each element based on the organization's requirements
Calculate the product's overall maturity score

Table 2. Open Source Maturity Model with default weightings (OSMM, Golden 2004)

	Phase 1: Assess element maturity				**Phase 2**	**Phase 3**
	Define requirements	Locate resources	Assess element maturity	Assign element score	Assign weighting factor	Calculate product maturity score
Product software					4	
Support					2	
Documentation					1	
Training					1	
Product integrations					1	
Professional services					1	

In a typical maturity assessment, score scale is from 1 to 10. The template in Table X is available in http://www.navicasoft.com/ as well as the example assessment for Drupal. Each step in the table is closely examined in the following.

Phase 1

Define organizational requirements for a particular element. This is a key step to assess the usefulness of a product for a particular organization.
Locate resources. Locating resources for an element is more challenging for open source products, but each chapter (in the book) offers a number of methods to identify that can assist an organization in implementing open source software.

Assess element maturity. Determining where the element lies on the maturity continuum – from non-existent to production-ready – lets an organization determine how likely the product will be to satisfy its requirements.
Assign element maturity score. Assignment provides the assessment of how well the product meets the organization's requirements. This score documents the consensus of the organization. The process of determining the score requires the members of the assessment team to resolve differences in perception, make concrete the reasons for their judgment, and come to a common agreement about the product element. The maturity score serves as an input into improving the element's maturity. Elements with low maturity score can be improved by the organization.

Phase 2

Apply product element weightings. Weighting allows each element to reflect its importance to the overall maturity of the product. Default weightings in OSMM are shown in Table X above. These weightings may be needed to adjust based on the specific needs of the organization. The only limitation is that the sum of maturity weightings must be ten.

Phase 3

Calculate the product's overall maturity score. After each element has been assessed and assigned a weighting factor, the overall product maturity score can be calculated. The elements scores are summed to give an overall maturity score on a scale of 1 to 100 that may be compared against recommended levels for different purposes, which vary according to whether an organization is an early adopter or a pragmatic user of IT.

Model #3: Qualification and Selection of Open Source Software (QSOS)

For a company, the selection to choose software as a component of its information system, whether this software is open source or commercially, rest on the analysis of the needs and constraints (technical, functional and strategic) and on the adequacy of the software to these needs and constraints. Atos Origin has conceived and formalized the QSOS method to ease this multi-faceted issue. They have made it available to all under the terms of GNU Free Documentation License. The method consists of four steps: definition, evaluation, qualification, and selection that are described in the following Table 3.

Table 3. Steps in QSOS evaluation

Step	Description
1. Definition	Constitution and enrichment of frames of reference used in the following steps.
2. Evaluation	Evaluation of software made on three axis of criteria: functional coverage, risks for the user and risks for the service provider (independently of any particular user/customer context).
3. Qualification	Weighting of the criteria split up on the three axes, modeling the context (user requirements and/or strategy set by the service provider).
4 . Selection	Application of the filter set up in Step 3 - "Qualification" of data provided by the first two steps, in order to proceed queries, comparisons and selections of products.

The first step, definition, includes defining the following sub-steps:
Software families, what functionalities needs to be included
Types of licenses, based on the three criteria: ownership, virality, and inheritance
Types of communities, five types identified to date: 1) Insulated developer, 2) Group of developers, 3) Organization of developers, 4) Legal entity, and 5) Commercial entity.

The second step, evaluation¸ comprises use of the identity card and the evaluation sheet. The identity card (ID card) consists descriptions of
General information (e.g. name, authors, references, licenses)
Existing services (e.g. documentation, numbers of contractual and training offers)
Functional and technical aspects (e.g. technologies of implementation, roadmap)
Overall synthesis (e.g. general trends and comments).

The evaluation sheet includes more detailed information than the ID card as it focuses on identifying, describing and analyzing in detail each evolution brought by the new release. The main phases are 1) scoring each criterion from zero to two, 2) functional coverage being determined by the software's family and proceeding the sub-steps in Definition, 3) estimating the risk from the user's perspective (e.g. intrinsic durability, industrialized solution, and integration). For more information, see the White Paper at http://www.qsos.org/download/qsos-1.6-en.pdf

The third step, qualification, defines filters translating the needs and constraints related to the selection of FOSS in a specific context. Filters can be set on ID card, functional grid (concerning required level of functionality), and perceived risks from the users and service providers perspective. Filters will be

defined in the O3S tool. Open Source Selection Software (O3S) is a single tool to apply the QSOS method in a coherent way. This tool is available to the community on the site http://www.qsos.org to coordinate creation, modification and use of QSOS evaluations.

The fourth and final step is selection which can be done by using strict or loose method. Strict selection is based on direct elimination as soon as software does not fulfill the requirements formulated in qualification step. Loose selection allows us to weight features and compare weighted scores against each other. The O3S tool enables the consultation of date related to a specific software and the comparison of software in the same family. This comparison is made by using weighted score patterns in a form of a radar chart.

Table 4. Summary of QSOS evaluation for MySQL 5.0

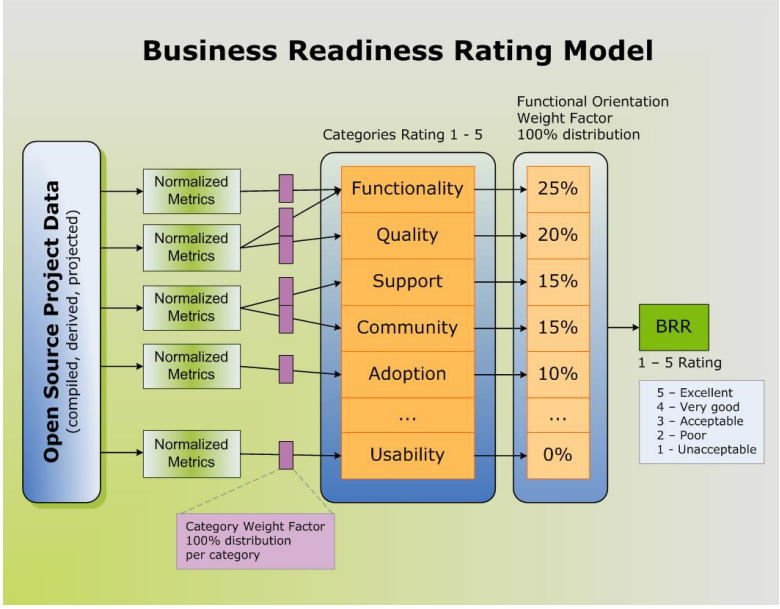
SECTION		Subscore	Overall
Generic			**68 of 90**
	Intrinsic durability	25/28	
	Industrialized solution	12/14	
	Packaging	19/24	
	Exploitability	3/4	
	Technical adaptability	3/6	
	Strategy	6/14	
RDBMS features			**27 of 42**
	SQL compliance	1/6	
	Classic SQL features	7/16	
	Security	2/2	
	Transactions	3/4	
	Other SQL features	14/14	
Advanced features			**3 of 10**
Tools			**6 of 8**
Overall MySQL rating			**104 (of 150)**

Model #4: Business Readiness Rating (BRR)

Business Readiness Rating™ (BRR) was proposed in 2005 as a new standard model for rating open source software. It is intended to enable the entire community (enterprise adopters and developers) to rate software in an open and standardized way. BRR is a community initiative that is being sponsored by Carnegie Mellon West Center for Open Source Investigation, O'Reilly CodeZoo, SpikeSource and Intel. The ultimate goal of BRR is to give companies a trusted, unbiased source for determining whether the open source software they are considering is mature enough to adopt. It helps adopters to assess which open source software is best suited to their needs and enables them to share findings with the community. It promotes use and adoption of open source software and may assist developers in creating and delivering software geared to enterprise use.

The calculation employed in the Business Readiness Rating model weights the factors that have proven to be most important for successful deployment of open source software in specific settings. Among these are functionality, quality, performance, support, community size, security, and others. The Business Readiness Rating model is open and flexible, yet standardized. This allows for broad implementation of a systematic and transparent assessment of both open source software and proprietary software.

Figure 1. Overall depiction of Business Readiness Rating Model (source: www.openbrr.org).



The model offers proposals for standardizing different types of evaluation data and grouping them into categories. To allow adoption of this assessment model for any usage requirements the software may have to meet, the process of assessment is separated into four phases: quick assessment filter, target usage assessment, data collecting & processing and data translation. First, a Quick Assessment rules in or out software packages and creates a shortlist of viable candidates. Second, it ranks the importance of categories or metrics, third, processes the data, and last, translates the data into the Business Readiness Rating. A software component's Business Readiness Rating is scored from 1-5, with one being "Unacceptable," and 5 being "Excellent." In the following sections, the Business Readiness Rating concept and a high-level overview of how to use the model will be presented.

Initial Filtering

To assess the business readiness of an open source software component, users may start by looking at several quantitative and qualitative properties of that component. During the initial Quick Assessment phase, a simple filter lets potential adopters quickly rule in or rule out software components with confidence. Several viability indicators to use as filters in this phase include:

- What is the licensing/legal situation of the software?
- Does it comply with standards?
- Are there referenceable adopters or users for it?

• Is a supporting or stable organization associated with the development efforts?
• What is its implementation language?
• Does it support internationalization and localization in your desired language?
• Are there third-party reviews of the software?
• Have books been published about the software?
• Is it being followed by industry analysts, such as Gartner or IDC?

The list of filtering criteria for Quick Assessment is by no means exhaustive. Users may and should add filters that are important for the particular software package or situation they are evaluating.

Metrics and Categories

After completing the Quick Assessment process, it is important to look at which metrics and categories to use for the in-depth assessment phases. Measurable properties of an open source software project are defined as metrics. To create a standardized Business Readiness Rating, the raw data of these metrics must be normalized. Quantitative metrics, such as the number of downloads of a software package, are relatively easy to normalize whereas normalization of qualitative metrics is more subjective.

Table 5. Twelve categories for assessing software

Assessment category	Questions describing the category
Functionality	How well will the software meet the average user's requirements?
Usability	How good is the UI? How easy to use is the software for end-users?
Quality	How easy is the software to install, configure, deploy, and maintain? Of what quality are the design, the code, and the tests? How complete and error-free are they?
Security	How well does the software handle security issues? How secure is it?
Performance	How well does the software perform?
Scalability	How well does the software scale to a large environment?
Architecture	How well is the software architected? How modular, portable, flexible, extensible, open, and easy to integrate is it?
Support	How well is the software component supported?
Documentation	Of what quality is any documentation for the software?
Adoption	How well is the component adopted by community, market, and industry?
Community	How active and lively is the community for the software?
Professionalism	What is the level of the professionalism of the development process and of the project organization as a whole?

A category rating is obtained by grouping together several metrics that measure the same aspects. How the rating in one category is calculated may differ from how another category is measured, but the results should use the same scale (1 to 5). One metric may contribute to several categories in different ways: for example, a release cycle of six months indicates a high level of community liveliness but a low level of stability.

Using the Model

The Quick Assessment phase and defining and ranking of metrics and categories according to their importance for the software's functional orientation leads us to the actions and steps taken in each assessment phase of the model to calculate the software's Business Readiness Rating.

Phase 1 – Quick Assessment

Identify a list of components to be evaluated.
Measure each component against the quick assessment criteria.
Remove any components that do not satisfy user requirements from the list.

Phase 2 – Target usage assessment

Category weights
Rank the 12 categories according to importance (1 – highest, 12 – lowest).
Take the top 7 (or fewer) categories for that component, and assign a percentage of importance for each, totaling 100% over the chosen categories.
Metric weights
For each metric within a category, rank the metric according to importance to business readiness.
For each metric within a category, assign a percentage of importance, totaling 100% over all the metrics within one category.

Phase 3 – Data collection and processing

Gather data for each metric used in each category rating, and calculate the applied weighting for each metric.

Phase 4 – Data translation

Use category ratings and the functional orientation weighting factors to calculate the Business Readiness Rating score.
Publish the software's Business Readiness Rating score.

2.3. Comparison of the models

When comparing the presented four models, we may notice that all models use methods like scaling, weighting, and normalization of metrics. It might be safe to summarize that no one can get away from these techniques. However, there exist differences in how these techniques are used or defined. Respectively, all four models define assessment areas. OSMM is strict in defining its assessment areas and enforce the utilization of all areas. BRR defines 12 assessment areas, and suggest the utilization of only seven of them. QSOS uses eight and they can also be tailored for each case.

In addition, QSOS and BRR differ in how exactly the metrics are defined. QSOS is not as precise as it only mentions a few metrics. BRR tries to be specific in defining metrics and the appropriate scales for them. It has been widely recognized that if a model is too loose, the assessment power of the model will be reduced. For example, a low quality product may obtain good rating if the model allows itself to be tuned to favor the product. However, this problem is immanent in all kinds of evaluations.

BRR White Paper (2007) states that such an evaluating model should include the crucial requirements of a good software rating model as being complete, simple, adaptable, and consistent (CSAC):

- Complete. The primary requirement for any product rating model is the ability of the model to highlight every prominent characteristic of the product, whether favorable or not. This is necessary to prevent that the rating for any product is never misleading.
- Simple. To gain wide acceptance, the model must be easy to understood and relatively easy to use. Furthermore, the rating and terminology should be customer friendly. However, the model's completeness takes a higher priority.
- Adaptable. Due to rapid changes in the software industry, any software rating model created today may be irrelevant in the future. During the conception stage, it is impossible to capture all future potential uses of the model. Therefore, we strive to build our model with adaptability in mind — and to keep it open. That way, when the model requires an extension, it will be easy to add one without much disruption of the current model.
- Consistent. The scales and ratings that the model produces should be consistent across the model's

different target uses. Comparable ratings for two software packages from two categories should signify equal business readiness.

We used these criteria in assessing the evaluation models presented previously. As can be seen in the following Table 6, the BRR model scores the highest overall result without any adjustments in weightings. Although the comparison can be regarded a bit superficial and illustrative, the BRR model gains the maximum points in the adaptability. As a summary, we will continue further by choosing the BRR model for our test case.

Table 6. Evaluation models assessed with CSAC criteria.

Criterion	**Weight**	**Optaros**	**OSMM**	**QSOS**	**BRR**
		Score	Score	Score	Score
Completeness	25%	2	3	4	4
Simplicity	25%	5	4	2	3
Adaptability	25%	4	3	3	5
Consistency	25%	3	3	3	4
Result		**3.5**	**3.3**	**3.0**	**4.0**

2.4. Evaluating Gnome with BRR model

We applied Business Readiness Rating at Nokia Multimedia. Our partner from Nokia was Mr. Quim Gil, who is responsible for Gnome-related matters at Nokia. Besides, being related to Nokia Quim Gil is a member Gnome Board, which makes high-level decision in Gnome.

Applying a maturity model in an industrial setting requires participation from industrial partner(s). The optimal setting for applying BRR would have been a case where one of our industrial partners would have been in a position for selecting an open source component. Since such cases are not too frequent, we were forced to select the next best option: applying BRR in an imagined setting.

Our frame story was that Nokia hadn't yet selected the desktop environment to its Internet Tablet, and now they should choose among the alternatives. In other words we try to make a time trip back to the days when Nokia Multimedia was selecting desktop system to their internet tablet. Our attempt was to apply the maturity model to Gnome.

As described in the previous section, applying Business Readiness Rating consists of four phases. This subsection describes the phases in out case study.

Quick Assessment Phase

BRR begins with the quick filtering phase, which allows the tester to quickly abandon the obviously unviable software components. Due to the fact that, Gnome is a well-known piece of software published under LGPL, it passes this phase easily.

Target Using Assessment Phase

The next task is set the weights to the seven most important categories. The category weightings are summarized in the following Table 7 and the justifications for these weight percents are opened up in the text below.

Table 7. Summary of category weightings

Category	Weight (%)
Architecture	20
Documentation	20
Adoptation	20
Quality	10
Community	10
Professionalism	10
Security	5
Support	5
Functionality	0
Usability	0
Scalability	0
Performance	0

Originally functionality was set as high as 15 %. However, our industrial partner was not willing to enumerate the functional requirements to the software. Therefore, we were forced to redistribute the weights so that functionality received 0%. The decision might seem rather strange. However, Quim’s opinion was that functionality is important, but if some piece is missing Nokia is able to code itself the needed one. On the other hand, discovering important functionality for a desktop was hard to the researchers, too. It is quite easy to list trivial features - like the system must support various input devises - but they are included in all desktops.

Architecture, documentation and adoptation were the most weighted categories with 20% for each. From a large corporation's point of view they form such a basis for a

software component, which can be tailored with respect to internal needs of the corporation. Architecture has a large impact to the rest of the categories. Having a decent architecture makes software evolution easier. The need for documentation is an obvious prerequisite for grasping the internals of the component. Having a large adoptation including other large companies ensures the continuity of the open source community. Besides, Nokia does not want to be the first-adopter.

Quality, community and professionalism received each 10% weight. According to Quim, the quality of the component core must be excellent - the rest can be fixed. The community produces the software Nokia doesn't want to develop. I.e. thriving community guarantees the future of the component. Besides, Nokia needs to interact with someone. Professional ecosystem is a strong criterion for choosing OSS in Nokia. The other companies make contributions which are usable for all parties.

Security and support are both given 5% of weight. We are dealing with a desktop environment, in which the security is not such a big issue. However, it is not meaningless. Justification for support the Nokia should not be the one who needs support, but the one who gives it to the end users. The usability category was given 0%, since such those problems can be fixed by Nokia. Another category receiving 0% was scalability, since it is strongly related to architecture, and thus, measuring it alone makes no sense.

Data Collection and Processing Phase

The actual data collection was carried out by the researchers with no assistance from the industrial partner. In this subsection we highlight measures from the most weighted categories.

Measures for Architecture
There are three measures in this category. Gnome’s unweighted ranking for this category is 5.

- "Are there any third party plugins?": Gnome goes easily beyond the limit for maximum score, which is as low as more than ten plugins. For example, site www.gnomefiles.org includes alone plenty of software for Gnome.

- "Public API / External Service": According to [OpenBRR.org] the purpose is to measure whether the product "allows for extensions via a public API, also

shows design for customization". Gnome is given the maximum score with respect to this measure, since all APIs are well documented in Gnome.

- "Enable/disable features through configuration": Due to the fact that Gnome is configurable even at runtime, it receives the maximum score also from this measure.

Measures for Documentation

The two measure related to documentation are "Existence of various documents" and "User contribution framework". The unweighted ranking for this category is 4.

- "Existence of various documents": This has been properly taken care in Gnome (5 points).

- "User contribution framework": In [OpenBRR.org] the justification for the measure is that the "best guides often come from user inputs and samples, as feedback from people who have used the products". We ranked Gnome to the middle ("People are allowed to contribute" ~ 3 points) in this measure. Gnome has a wiki and user forums at gnomesupport.org. If they were filtered by "experts" then the rank would have been the maximum.

Measures for Adoptation

The two measures for adoptation are "The number of books at Amazon" and "Reference deployment". The unweighted score for this category is 5.

- "The number of books at Amazon": This a clever and easy-to-calculate measure, as it is carried out by making a power search query at Amazon.com with query string "subject:computer and title:Gnome". In this case the number of books is 15, which means score 5.

- "Reference deployment": This measures that through a real-world deployment, that the software is scalable and tested in real use [OpenBRR.org] Naturally, Gnome is numerously adopted, but the number of users is not made public. Therefore, Gnome is given 3 points for this measure.

Data Translation Phase

The final task is simply to compute the Business Readiness for Gnome, and publishing the rating at BRR's www-page. Gnome's Business Readiness is 4.3.

2.5. Discussion and concluding thoughts

The Business Readiness Rating is an interesting opening towards a systematic evaluation of open source. The version of the method we used is not mature yet, after our trial it has been under further development. However, no new version has been published yet.

Finding the information required for evaluating the measures took roughly two to three working days. Moreover, we spent a half working day with our industrial partner. Using the BRR does not require special skills. The evaluation can be carried out by an engineer who is familiar with the application area of the software under evaluation.

One can observe similarities between software testing and BRR-like evaluation of OSS. In both the attention of the engineer is paid to small pieces of the system at a time, and then this small fragment is evaluated or tested. Similarly to the inability of testing to show that the software is error free, BRR cannot ensure the maturity of the software, but it can give us confidence when choosing open source – like when testing engineers find no hard flaws in a system makes the system trustworthy.

Unfortunately, we had no time to carry out the same evaluation we carried out for Gnome for some other desktop system. It might have given us a more elementary understanding of measuring OSS with BRR.

One problem we experienced during our trial was how to limit the system under evaluation. In other words, which plugin projects must be considered as being part of Gnome, and which are third party ones. This selection has a large impact to some measurements. For example, selecting a larger fraction leads larger amount of Gnome-related discussion groups. This, in turn, leads to more lively discussions and better measures.

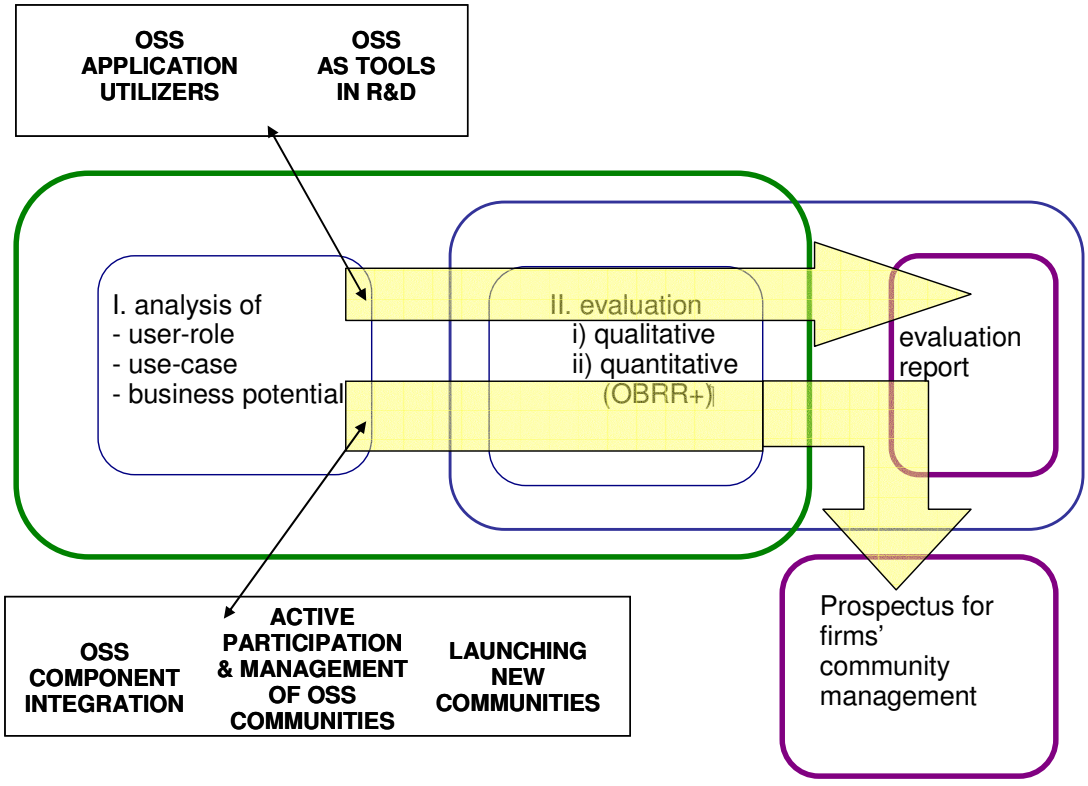
We were not comfortable for evaluating all the measures. Some measures required information that was so fuzzy, that their evaluations are simply based on educated guesses. Moreover,

the developers of some measures have had quite precise model of utilized software engineering processes and tools. When this model is not directly applicable to the software under evaluation, the evaluator simply has to stretch the measure to fit in her case. On the other hand, a big amount of measures were simple and easy to apply.

All in all, using a method like BRR is recommendable when a maturity of OSS is under evaluation. The value is not only in the final result the method produces, but the process itself. It gives a structured way for investigating the software product. The final number indicating the maturity does not reveal some risks of the software. On the other hand a small number definitely reveals that the product is not mature.

The lessons learned by using the existing evaluation methods have led us to believe that a two-step evaluation is necessary (see Figure 2 below). The user-role and intended use have a tremendous effect on how the software, the community and the interaction should be approached. Not only are communities different from each other, also the user-roles necessitate various types of analysis and involve different types of risks.

Figure 2. Evaluation process for different user roles



Consequently, the first step of the evaluation should consist of identifying the software to be used and, crucially, the user role of the company in question. Identifying and analyzing the user role is essential for asking the right kind of questions and identifying the possible bottle-necks and risks in the longer run. (For instance, the user-role dictates which sets of questions in the OBRR evaluation are relevant and how these sets should be weighted). By doing this it is also possible to assess the business possibilities of the use-case. So the questions asked in the first step are:

I. What software exits for the task in question and what communities are behind these software,
II. How will the software be used and what is the user role of the company (e.g., one out of the 5 presented above) and, finally,
III. What kind of added value is sought by the OSS use, such as time-to-market, outsourcing, cost savings?

Using open source software as a part of business has two sides, internal and external. Usually most of analysis focuses on external issues, for instance how to select the best piece of software or how to assess the viability of a particular community. However, internal issues may play vital role in succeeding implementation of open source. They represent the first phase of the evaluation, such as defining the business case.

Internal analysis should start with recognizing current and future needs. Questions that are useful in recognizing the reasons behind selecting open source software are, for example, the following:

- Analyzing time scale and urgency
  - How soon the output should be on market?
  - What is the overall life cycle of the output?
- Analyzing firms own resources and competences
  - What competences you need to a) select b) acquire c) maintain a software (this issue relates closely to outsourcing/purchasing)
  - How many resources you are able to invest for this issue?
- Analyzing the reasons to use open source software
  - Can you recognize your explicit and implicit motifs?
  - Why it is a strategic decision?
  - What are the main drivers?
  - What is the proposed use: are you going to use that particular piece of software in experimenting, piloting or production?
- Analyzing the status of relevant information

    - Do you know what you do not know?
- Analyzing the future
    - When the decisions are made, what consequences will follow?

An assessment task is about tradeoff between accuracy and time (i.e., money). Depending on answers on the questions above, one should make decisions what will be the needed level of information. An analytical and detailed approach may be too time or resource consuming when the software is just being experimented. Based on this assessment, a business model for particular business case can be designed (See e.g. Seppänen et al., 2007).

The second step in the analysis consists of both qualitative and quantitative evaluation of the software. The qualitative analysis is performed by answering a set of questions assessing the risks (legal, economical, cultural, social) connected to the communities in question. In addition, the license checker can be used to analyze the legal situation with regard to the code. Furthermore, the communities will be classified into the ideal types discussed above, so that suitable do's and do not's can be identified as the basis of the management framework. The quantitative analysis may be performed by an augmented version of the OBRR. We feel that especially in the case of high involvement with a community the questions provided in the OBRR need enrichment with regard to both socio-cultural and technological sustainability issues (see e.g. Helander & Antikainen 2007, Vainio et al. 2007).

For most cases of the two low-intensity user roles, application utilization and OSS as tools in R&D, we feel that the evaluation report produced by the qualitative and quantitative analyses will be enough. However, in the more intensive roles, from component integration to launching new communities, the evaluation report will be complemented by a prospectus for community management, consisting of practical do's and do not's, guides on best practices and long-term plans that help in organizing fruitful co-operation.

References

Golden, B. (2004). Succeeding with Open Source: Addison-Wesley.

Helander Nina, Aaltonen Timo, Mikkonen Teemu, Oksanen Ville, Puhakka Mikko, Seppänen Marko, Vadén Tere & Vainio Niklas. (2007). Open Source Software Management Framework. e-Business Research Center. Research Reports 38.

Helander, Nina. & Antikainen, Maria. (eds.) 2007. Essays on OSS Practices and Sustainability. e-Business Research Center Research Reports 36. Tampere University of Technology & University of Technology.

OpenBRR.org. (2005). Business readiness rating.

Optaros Inc. Retrieved August 20th, 2007. Accessed from http://www.optaros.com/en/publications/white_papers_reports/open_source_catalogue_2007.

QSOS. Retrieved August 17th, 2007. Accessed from http://www.qsos.org/.

Reidar Conradi, Amarjit Singh Marjara, Øivind Hantho, Torbjørn Frotveit, and Børge Skåtevik, "A Study of Inspections and Testing at Ericsson, NO", a rewritten edition of the paper presented at PROFES'99. Oulu, Finland, June 1999.

Seppänen Marko, Helander Nina & Mäkinen Saku. (2007). Business Models in Open Source Software Value Creation. In D. A. Becker (Ed.), Electronic Commerce: Concepts, Methodologies, Tools and Applications. New York: IGI Global Inc. pp.1103-1114.

Vainio Niklas, Oksanen Ville, Vaden Tere & Seppänen Marko. 2007. Elements of Open Source Community Sustainability. Poster at the OSS2007, 11-14 June. Limerick, Ireland.

3. Open Source Software Usage within Finnish Public sector - motivations to use IT, user acceptance and practical usability

Outi Grotenfelt
PhD student, Hanken
Lecturer, Helsinki Polytechnic Stadia
outi.grotenfelt@stadia.fi

Abstract

The maturity of Open Source Software (OSS) has reached a state where the software begins to be reckoned with also within public sector users. In Europe this has necessitated to several quite large bureaus changing to OSS tools instead of proprietary software (e.g. Germany, Norway, Italy, some states in South-America and Far East). In Finland the interest has been fairly low and only the first larger change has just happened in the Ministry of Justice with its 10.000 new OSS users. In the present study the total length of the project from the motivations, why and how to do the change, until the user acceptance and product usability has been followed up within a few public sector actors. To get also clear ideas and understanding, why OSS was not taken to use, a few actors representing that aspect were also included in the study.

Keywords: Open Source Software, OSS, motivation to use IT, software acceptance, and usability of IT tools.

3.1. Introduction

The definition of Open Source introduces software where the source code, in contrary to proprietary software, is available. Open Source Software has even got its nickname, OSS or F/OSS (Free/Open Source Software). The rights of OSS are protected by Open Source Initiative (OSI), which is a non-profit corporation dedicated to managing and promoting the Open Source Definition.

Voluntary individuals, who have been interested in supporting and contributing to a product or technology, program the additional software. This way of working was made possible and known by the Linux project, when Linus Thorvalds put out on the web his kernel to an UNIX-like operative system year 1991. Since that time there has been web-space for the code, chat-lists for the developers and co-workers and other possibilities in the Internet. A big part of ongoing projects are now working under the umbrella of SourceForge. There are available the basic tools for an Open Source Project.

There are several very successful OSS products and technologies, including software, computers, devices, technical formats, and computer languages. Linux is perhaps the best known of the products. Other well-known ones are Apache, MySQL, BIND, sendmail, Mozilla and OpenSSL. There are also open programming tools like PHP, Perl and GNU. Several companies have made a remarkable contribution and based their business model on open source products. Among them there are IBM, Apple, Novell, HP, Sun, Red Hat and Sleepycat just to name some.

The significance of OSS is that the software is released early and often. To the opposite of proprietary software, where releases come at first when the product is ready, OSS products are also tested on the net and the raw versions of the code are available. This became a problem with Linux in times and there was made a decision on it, that the versions with certain style of version numbers are stable, company-usable and the developing versions in between were numbered differently.

The reasons to use OSS programs have been several and the claims have been that OSS programs are more reliable, secure, stable and so on. Proper research, weather these kinds of comments are really accurate are lacking but the reputation seems to be obstinate.

The other usual claim on OSS is, that it is “gratis”, the programs are free and that’s why it should be used all over (and specially much by the administrative bureaus in companies and public sector). However nowadays, when there is always already some legacy ICT system with which the eventually new programs must be able to co-operate, the cost of software programs can be a smaller part of total costs (including integration) than the average human being expects it to be.

3.2. Research questions

This study has been arranged around following research questions:

- In the change process to F/OSS could some special reasons or behaviour be observed that contribute to the change and can be seen and analyzed?
- How were the F/OSS possibilities investigated, introduced and grounded? What was done to motivate the users to accept the new software?
- How to measure of the usability of F/OSS?

3.3. The different information technology acceptance models – modification towards expectation acceptance model (EAT)

3.3.1. Theory of Reasoned Action (TRA)

Theory of reasoned action was introduced in the late 80's (Sheppard et al. 1988). The theory is drawn from social psychology and it is one of the most fundamental and influential theories of human behaviours (Venkatesh et al. 2003). The basic components of TRA are 1) *attitude toward behaviour*, where an individuals positive or negative feelings are measured when performing the target behaviour, and 2) *subjective norm*, where the attitude from the close neighbourhood of an individual is considered to have an affect on behaviour. However there were 10 items found in this model affecting satisfaction: accuracy, reliability, timeliness, assistance, adequacy, accommodation, communication, access, cost and environment.

In this study *attitude toward behaviour* is considered to be an important factor wears the *subjective norm* is less important: IT usage is so everyday life for all people nowadays and specially on all working places that the attitude of ones working neighbourhood can not be measured. Everybody just must use the computers; the opposite possibility does not exist any more. However the tools used can be very sensible: some users are very keen on using the same they have used before or want to have specifically certain (proprietary) software.

3.3.2. The technology acceptance model (TAM)

TRA was implemented for technology acceptance by Davis (Davis et al. 1989). He tailored the model to predict information technology acceptance and usage on job. TAM excludes the attitude construct in order to better explain intention parsimoniously (Venkatesh et al. 2003). The basic components in TAM are *perceived usefulness*, which describes the degree to which a person believes that his/her performance is depending on a particular system, and *perceived ease of use*, where the system usage is convenient and easy.

The *subjective norm* has been added to some later construction of TAM (TAM2, Venkatesh & Davis 2000). Its background is in TRA and TPB.

The three important insights in the Davis TAM research (Davis et al. 1989) where

1. People's computer use can be predicted reasonably well from their intentions.
2. Perceived usefulness is a major determinant of people's intentions to use computers.
3. Perceived ease of use is a significant secondary determinant of people's intentions to use the computers.

So the behavioural intention (BI) was the major determinant of usage behaviour, the other factors have an indirect impact by influencing BI.

TAM has been tested in many different kinds of research settings: both for software and hardware/system. In the very first research there was a WordOne program that a bunch of MBA students tested (Davis et al. 1989) through this specific *behaviour* (usage) toward a specific *target* (WriteOne) within a specific *context* (the MBA program). Later on TAM model has been used for example explaining intranet use (Horton et al. 2001).

In this research TAM gives a stable theoretical background concerning acceptance of the new software system. It has not been used earlier to measure open source software (OSS) acceptance in companies nor within public administration. The way of describing different important factors concerning acceptance of IT is quite simple in TAM and due to that relatively simple to use. Its "decoration factor" is also low, so it can well be used for research on acceptance of OSS neighbourhood environment in FMI (the Finnish Meteorological Institution).

3.3.3. Theory of planned behaviour, TPB

TPB extended TRA by adding the construct of perceived behavioural control (Venkantesh et al. 2003). In this model there is added to the intention and behaviour a perceived behavioural control. This theory was tested by Ajazen (1991) in several settings. Also many others have tested TPB and the results widen the understanding of individual acceptance and usage of many different kinds of technologies. There is also a related model for this, Decomposed Theory of Planned Behaviour (DTPB). This model is similar to TPB in terms of predicting intention but differs from it (and is similar to TAM) in "decomposing" the attitude.

This theory was not seen as adding many new perspectives to this study so its particualr aspect, perceived behavioural control, was not included on the grounds of not having an important impact to the results.

3.3.4. About Information Technology Use / the Construction of Beliefs

This model (Lewis et al 2003) is quite a lot younger than the previously introduced models. So the overall usage situation of computers has changed dramatically during the last decade. For example today a firm or bureau cannot work without computers.

When some new software or hardware is taken to use this model groups institutional, social and individual factors, that affect on the usefulness of a new system. Within institutional factors the most important are, the management comitment to the new system, both at the top and local levels. The impact of social factors to one user depends on departmental and professional peers, supervisors and senior leaders and also the informal circles workers can have on their working places. The individual factors consists on the personal innovativeness with technology and from the computer self efficiency.

All these different factors affected on perceived usefulness, which was measured in this model, in combination with ease of use.

3.3.5. Unified Theory of Acceptance and Use of Technology (UTAUT)

This theory has been developed from eight different basic models that have been empirically tested and compared. Those models were theory of reasoned action (TRA), the technology acceptance model (TAM), the motivational model (MM), the theory of planned behaviour (TPB), a model combining the technology acceptance model and the theory of planned behaviour (TPB/DTPB), the model of PC utilization (MPCU), the innovation diffusion theory (IDT) and the social cognitive theory

(SCT). The resulting theory, unified theory of acceptance and use of technology (UTAUT), consists mainly of expectancies from performance, effort, social influence and facilitating conditions. When these are influenced by gender, age, experience and voluntariness of use, the systems behavioural intention and use behaviour can be expected (Venkantesh et al. 2003).

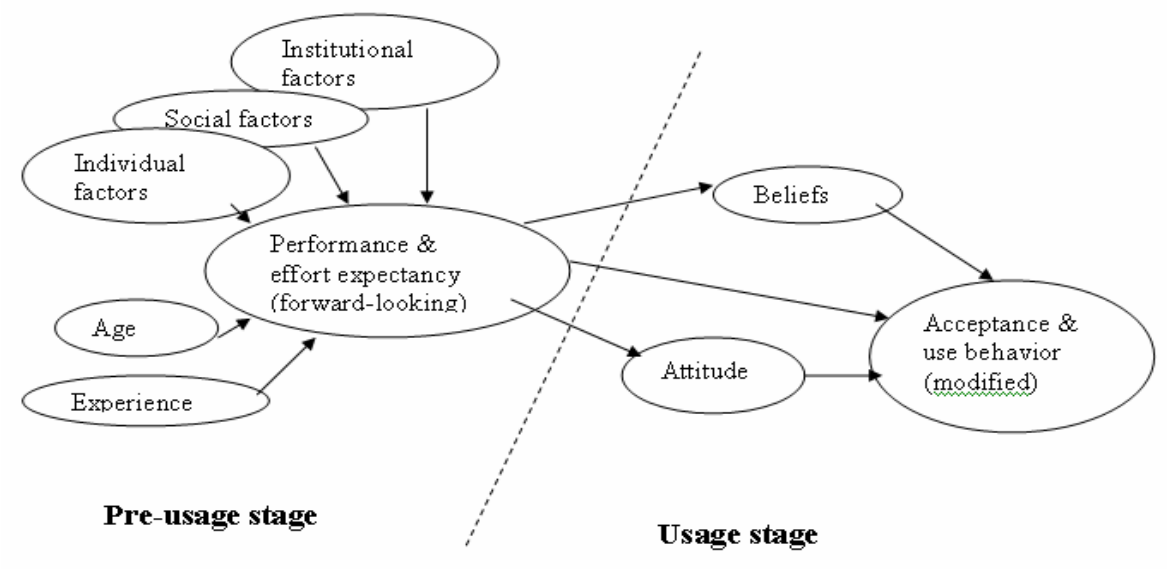
3.3.6. Expectation-disconfirmation theory (EDT)

The last model taken as groundings in this study is the expectation-disconfirmation theory (EDT), (Bhattacherjee, A. and Premkumar G. (2004)). There is pre-usage and usage stages separately but affecting to each other. In the pre-usage stage the communication and other antecedent's effect on forward-looking beliefs and attitude. When the users reach the status of users, their disconfirmation and satisfaction together with modified beliefs and intentions build up their modified attitude on the new ICT system.

3.3.7. Result: Expectation acceptance model (EAT)

The result of the literature study and the first findings of the cases done leads to thenneed ot develop the earlier model where the time function of the implementation process is emphasized. Therefore a model (EAT) where the pre-usage stage and the usage stage are considered relevant is proposed. This model will be evaluated in the empirical part of the study. The most important factor in the cases researched were the institutional factors (pre-usage stage): the top and local management commitment. Social factors and specially peers in the bureau proved to influence a lot to the overall attitude. From the individual factors the personal innovativeness with technology seems to be much more important than age or experience: curious mind makes the new users to try and play with new systems and get familiar with them. The change could be made a lot easier by informing workers within a longer period of time and by taking a fairly long decision period, when the new system could be well introduced.

Figure 1: The expectation acceptance theory (EAT)



The modified (usage stage) user acceptance and use behaviour were formed through beliefs and attitude consisting the forward-looking performance and effort expectancy.

3.4. Research methodology

3.4.1. Study context and Sample

This research has contained 3 different case studies. Their order has been the opposite to the presentation in the study so that the starting case was the web-questionnaire within Finnish Meteorological Institution (FMI, CASE 3, performed 5/2006) even if it will be introduced to the last. The second case (CASE 2, performed during spring 2006) was interviewing users within the topics around acceptance and usability. This part of study is postponed and will still get a time perspective with a second phase of interviews. In this paper CASE 2 and 3 are introduced together. The last part (CASE 1, performed spring 2007) is under construction and contains the interview research on reasons and motivations for choosing (or not choosing) OSS.

Participants in these three different phases were collected as follows: in CASE 1 there were workers in different public agencies (9 interviews). The participants were working in the Ministry of Finance (FM), the Ministry of Justice (MJ), FMI and city of Turku. The interview case of usability and acceptance, CASE 2 contained workers in the Meteorological Institution of Finland. The web-questionnaire from FMI contained 185 answers from FMI and 15 workers from the Ministry of Justice (this part is aimed to be renewed with a larger group).

CASE 1: Reasons and motivations of choosing OSS

The used data were collected from the FM from one interview from a quite central person within the ministry. Unluckily other persons could not be interviewed. The other data consists of the statistical reports FM gives out each year (years 2001-2006).

In the MJ three persons were interviewed. One of them were the leading organizer for the OpenOffice.org (OOo) change they just have made, the other were a trainer educating the users to the new system and the third had been writing the new Ooo manuals. Also the minutes of meetings (8 pc), where the decision had been developing to its final stage, were available. The pilot report and OpenOffice user manual and questions and answers written in the bureau were also available in this research.

The FMI interviews contained the IT director and two IT workers, one with technical background and the other without it. Also internal information and an engineering thesis clarifying the starting point situation with OSS change were available.

From the city of Turku their managing person concerning the try to change to OSS use were interviewed. The other person from there is working as a trainer at lower university level. Also a Conference presentation (2002), the final report on OO & Linux suitability (2001) and a report: OSS use in the public administration (2003) was used in getting the idea of their process.

CASE 2: Research on FLOSS product acceptance and usability

A wide questionnaire was used to get the users attitudes of acceptance and usability. Until now there has been 6 interviews made and there is a plan to get some 6 more interviews during spring 2008.

CASE 3: Research on FLOSS product acceptance and usability

A web questionnaire containing some 23 questions was sent out to approximately 400 users (where some of the addresses were old) and 185 answers were collected. The same formulae were also sent to JM, but only to a group of 15 of their first stage testers (who also were chosen by the managers). This part cannot be considered as relevant data but is though an interesting case.

3.4.2. Quality of the research

The public administrative offices in motivation research were selected to a few players in the field. Some of them were known as OSS users (FMI, MJ) and the others were known as non-users (FM, city of Turku). The persons interviewed were in central posts while some research or decisions had been made. At least one name could be located in that way and also the located person were available in all places. OSS research and selection processes in different agencies are very much based on networks and that gave also some ideas of possible interesting persons. Also these central persons gave some ideas who else could be interviewed from that office and mostly these interviews were also conducted. The situation were not so lucky in FM and Turku, the first because of the work load and schedule problems of the other possible interview and from Turku because the process had happened such a long time ago and other workers participating the research process with OSS had already changed they workplaces.

Acceptance and usability of OSS systems were tested in FMI. This was mainly because they were already users and they had quite a lot of experience of OSS systems. And also the top managers were positive towards this research. The sample with the web-questionnaire was large, approximately 350 mail addresses and the respondent rate was 52,9% . The web-questionnaire consisted of questions with answer possibilities (yes/no/no opinion or of the same/nearly same/a little opposite/very much opposite/no opinion) where scaling were close to Likert scale with the possibility of not taking any opinion. There were also several open questions, which were widely answered and gave a good insight of the opinions even with only a web-questionnaire.

The interviews done on acceptance and usability research are still quite few and represent mostly technical and clerical staff. This can though give relative good insight of the opinions among users.

3.5. First founding's in the research

3.5.1. Interviews on reasons

The Ministry of Justice has centrally administrated IT and they started to have updating problems 2004-5. They started a testing project and considered that also OSS programs were one possibility for the office tool. The operating systems were not insisted upon to be changed: Windows were to be used also after the renewal process. MJ wanted to offer all users the same standard desktop; mostly with OpenOffice.org but also MS Office to guarantee service to all co-operation partners.

The overall decision process was slow and there were much testing and information to final end-users. Also the decision to deliver OOo on a CD-rom or in a memory stick to the becoming end-users under the decision process for training and trial seemed to be wise. The final conclusion came in the end of 2006 and from the beginning of 2007 OpenOffice has been installed to all users (altogether 10.000) and a lot of training has also been done. MS Office was decided to be left to 1.500 users because of the service checking.

The change were possible because of the centrally administrated IT. The whole organization uses the same kind of standard desktop and as long as the co-operation is only in-house, even the usage of OOo does not cause any problems. OOo have been tested to be usable, compatible and the open formats it uses for document saving preclude the organization of a lock situation of one software vendor. Also price played an important role.

The process within the Finnish Meteorological Institution was different. They have been OSS users from the very beginning; they have users, who have had Linux in use from the version 0.001 or something like that. But their aim with the change process were to get a standard desktop to the researchers and get all helpdesk work, as backups and such, a lot easier and more possible to do. Their policy has been that each researcher can get any programs they want to for their use.

FMI tried though do a double change: to change both as much of the software as possible to OSS and also to change mostly PC's to thin clients. The project started by testing two different platforms, SunRay and LTSP Linux-thin client. The problem has been that all users can have chose quite voluntary the new thin client system if they have wished. And the other problem has been that the thin client hardware has not been functioning as sure as were expected. And the third problem was the merging OSS and thin client together and when problems occurred, no progress has been made on any of the changes. Open software code is an important thing in several fields of research, especially within space research. As was considered, "*black-boxes can't be sent in satellites*", you have to know each programming code line.

The interesting opinion from a technical worker was though, that when the process started, the researcher intention was "***if** I have to change to that*" and by the time of the interview (about a year later) the intention had changed to "***when** I have to change*..." The stability problems with the hardware have to be dealt with and to get further with the project; it could be a good decision to separate software and hardware changes. The lack of proper project plan, the prolonged timing and several basic problems with the desktop hardware has aroused a lot of frustration among end-users.

The Ministry of Finance is a top-organization for different agencies that can be counted to 250-800 bureaus with approximately 120.000 workers and 160.000 PC's. Their working field consists about 4.800 different programs. However a lot of their work is sharing documents. For those purposes there have been made a lot of programs relaying on MS .doc-format. So the change from MS .doc to any other format would be terribly expensive. However, also the lack of political commitment among politicians in Finland is a great break to the development towards OSS usage.

The city of Turku is a typical municipal setting. For their purposes there is no difference in the origin of their software, the most important things are usability, compatibility and price. Tax money should be effectively used. In the city the change possibility to OSS programs were researched in the beginning of 2000. The result was negative and the reasons for concluding to that were partly lobbying against OSS alternative and partly lack of top management commitment, too small group working on the case and immature programs available on OSS sector by that time. To be the first mover on a new niche is always tricky.

In educational setting the reason can merely be that “de facto” standard within business is still MS tools and that is a grounding to use them with students. Companies can buy new workers, who are available to use those tools effectively.

Conclusions

Within these public organizations it seemed to be a very important fact that the change towards OSS usage needs to have centrally administrated IT. Also the management support and commitment seemed to be of great importance from the beginning to the end of the project. For nearly all of the administrative players it was not an issue, weather software is open or closed or proprietary, usability more important. The only exception was FMI with its space research. The problem can though be, that in some bureaus there were enthusiasts: workers who really had an emotional relation towards software and specially OSS. However also the educational level of users seemed to be irrelevant, both low and high-educated users could be good users of OSS or oppose their usage.

3.5.2. Web-questionnaire & interviews on acceptance and usability

Web-questionnaire & interviews/FMI

First in the web-formulae the usual facts of gender, age and educational level were asked. In FMI nearly 70% of the workers are men, about 60% over 40 years old and some 65% has at least MSc (or higher) education. They work mostly as specialists and have a long experience with computers.

The ability to use Windows is excellent and also knowledge in Linux is much better than average in bureaus and firms: some 55% knows Linux well enough for their own usage. The typical programs used daily or weekly by everybody are web-browsers and mail and word processing. Calendar, spreadsheet and presentation graphics are fairly important to half of the workers.

Knowledge of OSS programs within FMI is high; nearly 80% of the workers have a basic knowledge of Open Source. The overall opinion is also relatively positive as half of employee was of the opinion that there can be found a suitable program within OSS programs for their special usage. However the users in FMI consider, that there programs should be easy to use, operate also in older computers and should be adjustable for their own usage.

OSS programs are considered as safe, usable, easy to use, of good quality, are effective in code and cheap (or cost effective; over 60% were of the same or nearly of the same opinion).

Web-questionnaire/ MJ

In the Ministry of Justice, the problem is a very small sample, only 15 respondents. The age distribution was quite alike FMI but gender and educational level were nearly opposite. Even their experience of using computers was as in FMI. One half of the respondents were working with IT tasks and the other half with clerical work, which does not congruent with the working force at MJ.

Employees at MJ knew Windows quite well but their knowledge on OSS were superficial. The important programs for them are mostly web-browser, mail and word processing. Some uses calendar or spreadsheet counting / presentation graphics. From the OSS world they were familiar with OpenOffice.org and had mostly some idea of Linux and Mozilla. More special programs were unknown. However the overall oppinion towards OSS were quite positive and nearly everybody thought, that they would certainly find suitable program for their needs.

Employees seemed to be confident with the functionality and ease of use of OSS programs. Those programs were seen to be safe, usable, easy to use, of good quality and effective source code and also cost effective.

Conclusions/FMI & MJ

The interesting observation in this research has been that neither has gender or age nor educational level too much to do comparing to the attitudes or ability to use certain software among different users. The more important factors are inquisitiveness towards new equipment/software, managerial support and perhaps some training how to use it.

Possibilities to start using OSS programs depends very much on commitment and also from the earlier history. In the Ministry of Justice MS Office programs have never had a hegemonial state and they have a centrally administrated IT so they really had a possibility to change. Also this web-questionnarie showed well, that a slow process with proper project planning and testing and information to the becoming users lowers the opposing opinion. Their very good knowledge on OpenOffice during this web-questionnarie were possible only because they had been testing it already. Even the slow motion of the whole project gave the tool (OpenOffice) the time to get the following, more mature and user friendly version with also finnish language proof-reading tool soikko to get ready.

3.6. Discussion

The whole project has been extremely interesting. The research follow-up period started 2002 and has continued from that time. The difference seen in OSS programs, their maturity and usability compared to the state in the beginning of the project is huge. The attitudes have got to a more reasonable level from the hype to reality. The important things in choosing new software are nowadays usability, compatibility and together with that also price. Total cost is much closer counted also for programs “available for free/gratis”; as Stallman considered from OSS: *“there is no such thing as free beer”*.

References

Adelstein Tom, 2004, The Open Source Dilemma for Governments, Linux Today, http://www.linuxtoday.com/infrastructure/20040106009260PSWPB, last visited 12/2007

Bailey James E, Pearson Sammy W, 1983, Development of a Tool for Measuring and Analyzing Computer User Satisfaction, Management Science, vol 29 issue 5 pp. 530-546

Benkler Yochai, 2002, Coase's Penguin, or, Linux and The Nature of the Firm, The Yale Law Journal, vol 112

Bessen James, 2005, Open Source Software: Free Provision of Complex Public Goods, http://www.researchoninnovation.org/opensrc.pdf, last visited 12/2007

Bhattacherjee Anol, Premkumar G, 2004, Understanding Changes in Belief and Attitude Toward Information Technology Usage: A Theoretical Model and Longitudinal Test, MIS Quarterly, vol 28 no 2, pp. 229-254

Bizer Jürgen, Schröder Philipp J H, 2005, The Impact of Entry and Competition by Open Source Software on Innovation Activity, International Busines Section, Aarhus School of Business, Workin paper 2005-12, ISBN 87-7882-106-1

Bonardi Jean-Philippe, Warin Thierry, March 2007, Open Source Software Development,

Innovation, and Coordination Costs, http://www.middlebury.edu/services/econ/repec/mdl/ancoec/0701.pdf, last visited 12/2007

Davis Fred D, Bagozzi Richard P, Warshaw Paul R: User Acceptance of Computer Technology: A Comparison of two Theoretical Models, Management Science, vol 35, no 8 August 1989

Fitzgerald Brian, Kenny Tony, 2003: Open Source Software can Improve the Health of the Bank Balance – The Beaumont Hospital Experience, http://www.netproject.com/docs/Beaumont.pdf, last visited 12/2007

Franck Egon, Jungwirth Carola, 2002, Reconciling investors and donations – The governance structure of open source, No 8, Working Papers from University of Zurich, Institute for Strategy and Business Economics (ISU), www.iou.unizh.ch/orga/downloads/EGOS2002/FranckJungwirth.pdf , last visited 12/2007

Frishberg Nancy, Dirks Anna Marie, Benson Calum, Nickell Seth, Smith Suzanna: Getting to Know You: Open Source Development Meets Usability, CHI 2002, April 20-25, 2002, Minneapolis, Minnesota, USA; ACM 1-58113-454-1/02/0004

Goode Sigi, 2004, Something for nothing: management rejection of open source software in Australia's top firms, Information & Management, vol 42 is 5 July 2005, pp. 669-681

Hedberg Henrik; Iivari Netta; Rajanen Mikko; Harjumaa Lasse, 2007, Assuring Quality and Usability in Open Soruce Software Development, First International Workshop on

Emerging Trends in FLOSS Research and Development, 20-26 May 2007, http://cross.lincoln.ac.uk/floss2007/papers/02floss-Assuring%20Quality%20in%20Open%20Source.pdf, last visited 12/2007

Heijden Hans vd, 2004, User Acceptance of Hedonic Information Systems, MIS Quarterly, vol 28 no 4, pp. 695-704

Horton Robin P, Buck Tamsin, Waterson Patrick E, Clegg Chris W, 2001, Explaining intranet use with the technology acceptance model, Jorunal of Information Technology, 16, 237-249

Holzinger, Andreas: *Usability Engineering for Software Developers, Communications of the ACM (ISSN 0001-0782), Vol. 48, Issue 1 (January 2005), 71-74*

C.A. Kenwood, 2001. "A Business Case Study of Open Source Software," at http://www.mitre.org/work/tech_papers/tech_papers_01/kenwood_software/kenwood_software.pdf, accessed 21 September 2005.

Klang Mathias, 2005, Free software and open source: The freedom debate and its consequences, First Monday 10/3

Legris Paul, Ingham John, Collerette Pierre, 2003, Why do people use information technology? A critical review of the technology acceptance model, Information & Management, 40, 191-2004

Lerner Josh, Tirole Jean, 2001, The open source movement: Key research questions, *European Economic Review*, 2001, vol. 45, issue 4-6, pages 819-826

Levesque Michelle, 2004, Fundamental issues with open source software development, First Monday, 9/4

Lewis William, Agarwal Ritu, Sambamurthy V, 2003, Sources of Influence on Beliefs about Information Technology Use: An Empirical Study of Knowledgr Workers, MIS Quarterly, vol 27 no 4 pp. 657-678/December 2003

Luthiger Benno, Jungwirth Carola : Pervasive fun, *First Monday,* volume 12, number 1 (January 2007),URL: http://firstmonday.org/issues/issue12_1/luthiger/index.html

Nichols David M, Thomson Kirsten, Yates Stuart A, 2001, Usability and open-source software development, http://www.cs.waikato.ac.nz/~daven/docs/oss.pdf, last visited 12/2007

Nichols David M, Twindale Michael B, 2003: Usability and Open Source Software, First Monday, 8(1)

Nielsen, Jakob (1994), *Usability Engineering*, Morgan Kaufmann Publishers, ISBN 0-12-518406-9

Oz Effy, 2005, Information technology productivity: in search of a define observation, Information & Management 42, pp. 789-798

Raymond Eric S, 1998, Homesteading the Noosphere, First Monday 3/10

Schmidt Klaus M, Schnitzer Monika, 2003, Public Subsidies for Open Source? Some Economic Policy Issues of the Software Market, Social Science Research Network,

http://papers.ssrn.com/sol3/papers.cfm?abstract_id=395461, last visited 12/2007

Shimizu Hiroyuki, Iio Jun, Hiyane Kazuo, 2004, The realities of Free/Libre/Open Source software developers in Japan and Asia, First Monday 9/11, FLOSS at large: Selected papers from the FLOSS workshop at 4SEASST joint conference, Paris, 25-28 August 2004, http://www.firstmonday.org/issues/issue9_11/shimizu/

Shneiderman, Ben: *Software Psychology*, 1980, ISBN 0-87626-816-5

Smith K T, Coventry L. "Usability Tools and the Design Process" Ergonomics Society Conference 2002

Soares Marcus Vinicius Brandão, 2004: Reducing transaction costs in information infrastructures using FLOSS, First Monday 9/11, FLOSS at large: Selected papers from the FLOSS workshop at 4SEASST joint conference, Paris, 25-28 August 2004

Venkatesh Viswanath, Davis Fred D: A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies, Management Science, 2000, vol 46, no 2 February 2000, pp. 186-2004

Venkatesh Viswanath, Morris Michael G, Davis Gordon B, Davis Fred D: 2003, User Acceptance of Information Technology: Toward a Unified View, MIS Quarterly, vol 27 no 3, pp. 425-478

West Joel, O'Mahoney Siobhán, 2005, Contrasting Community Building in Sponsored and Community Founded Open Source Projects, proceedings of the 38th International Confrence on System Sciences, Hawaii

Wheeler David A, 2004, Why Open Source Software/Free Software (OSS/FS)? Look at the Numbers!, http://www.dwheeler.com/oss_fs_why.html, updated 2007, last visited 12/2007

Zain Mohammed, Rose Raduan Che, Iskandar Abdullah, Masrom Maslin, 2005, The relationship between information technology acceptance and organizational agility in Malaysia, Information & Management 42, pp. 829-839

http://en.wikipedia.org/wiki/Usability

References to Finnish research material:

Ilmatieteen laitos: Sinisalo Pertti: Thin Client -arkkitehtuurin pilottiprojekti, projektisuunnitelma, 28.1.2005

Ilmatieteen laitos: Pantsar Ari: Linux-päätteet, Käyttäjän pikaopas, versio 1.0.2, 13.2.2006

Oikeusministeriö: Tietohallinnon yhteistyöryhmän kokouksien pöytäkirjat (niiltä osin, kuin ne liittyvät OpenOffice.org käyttöönottopäätöksiin) 2/2005 (22.2.2005), 8/2005 (25.10.2005), 1/2006 (25.1.2006), 2/2006 (7.3.2006), 3/2006 (28.3.2006), 4/2006 (2.5.2006), 5/2006 (29.5.2006), 6/2006 (4.9.2006), 7/2006 (26.9.2006), 8/2006 (24.10.2006)

Oikeusministeriön toiminta ja hallinto 2005:4: Oikeusministeriön hallinnonalan toimisto-ohjelmaselvitys, Lotus Smart Suite,

Microsoft Office, OpenOffice.org, Oikeusministeriö, Helsinki, 2005

Oikeusministeriön toiminta ja hallinto 2006:29: Oikeusministeriön OpenOffice- pilotoinnin loppuraportti, Oikeusministeriö, Helsinki, 2005

Valtiovarainministeriö:

Tietoja valtion tietohallinnosta ja tietotekniikasta 2000, ISBN 951-804-245-4, Edita Oyj, Helsinki 2001

(vastaavat vuosikirjat käytössä kaikilta vuosilta)

Tietoja valtion tietohallinnosta ja tietotekniikasta 2006, ISBN 978-951-804-719-6 (tai 720-2), Edita Prima Oy, Helsinki 2007 (vastaavat vuosikirjat käytössä vuodesta 2000)

Sähköisten palveluiden ja asioinnin tietoturvallisuuden yleisohje, 4/2001, ISBN 951-804-240-3

Valtion viranomaisen tietoturvallisuustyön yleisohje, 1/2001, ISBN 951-804-211-X

Valtion tietotekniikan rajapintasuosituksia, 27/2001, , ISBN 951-804-255-1, http://www.vn.fi/vm/julkaisut/työryhmämuistiot/index.html

Kysely StarOffice/OpenOffice toimistosovellusten testaajalle (Liite 1), palautus 1.2.2002 mennessä & Käyttäjien kokemuksia ja havaintoja StarOffice/OpenOffice-toimistosovellusten testaus valtionhallinnossa (Liite 2), raporttiin StarOffice/OpenOffice-toimistosovellusten testaus valtionhallinnossa – käyttäjien kokemuksia. Valtiovarainministeriö. Avoimen koodin projekti 2002.

4. Avoimen lähdekoodin tutkimuskysymyksiä

Juho Lindman
Tutkija, Tietojärjestelmätiede,
Helsingin kauppakorkeakoulu
juho.lindman@hse.fi

Abstrakti

Avoimen lähdekoodin tutkimuskenttä on siirtymässä tutkimusalan oikeuttamisesta kehittämistavan hyötyjen siirtämiseen organisaatioihin. Tätä muutosta vauhdittavat ohjelmistoalaa koskeva muutos kohti alihankintaa, yleinen muutos kohti liittoumia sekä muutos yhteistyöstä avoimen lähdekoodin kehitysyhteisöjen kanssa. Eräs tapa hyödyntää avointa lähdekoodia on pyrkiä rakentamaan kehittäjäyhteisöjä yrityksen sisään, toinen on rajoittaa muuten lähdekoodin näkyvyyttä sopimuksin. Vaikuttaisi siltä, että organisaatioiden ja avoimen kehittämisen törmääminen johtaa useissa tapauksissa kehittämisen avoimuuden vähenemiseen. Tämän puheenvuoron tarkoituksena on kartoittaa tutkimuskentän muutosta ja esittää muutamia sitä koskevia kriittisiä havaintoja.

4.1. Johdanto

Avoimen lähdekoodin ohjelmistojen käyttö ja kehittäminen ovat yleistyneet yrityskäytössä. Osaltaan siksi tarkastelu on siirtynyt organisaatioihin avoimen lähdekoodin hyödyntäjinä. Tätä muutosta tutkimuskentällä kuvaa muun muassa Brian Fitzgeraldin (2006) tutkimuskenttää ohjaava käsite OSS 2.0, joka pyrkii kuvamaan avoimen lähdekoodin hyötyjä organisatorisista lähtökohdista.

Käytännössä muutos tutkimuskentällä merkitsee myös monia muutoksia tapaan, joilla ohjelmistoja koskevia asioita on perinteisesti ajateltu. Ohjelmistolisenssejä on myyty suunnilleen niin kauan kuin se on ollut teknisesti mahdollista (Riepula, 2008). Avoimen lähdekoodin lisensointitavat ovat kuitenkin asettaneet kyseenalaiseksi ohjelmistojen levitysmäärään perustuvat hinnoittelumallit (Välimäki 2005). Ohjelmistotuotteen ja palvelun rajankäynti on johtamassa jälkimmäisen roolin korostumiseen (Cusumano, 2004).

Samaan aikaan tutkimuskentän muuttuessa myös ilmiö on muuttunut. Spontaanit organisoitumisen muodot, vapaaehtoisuuteen perustuva ohjelmistotyö ja avoimeen jakamiseen perustuva organisaatiokulttuuri ovat saaneet haastajikseen vakiintuneiden yritysten toimintatavat. Nämä organisaation liikevoiton ja kilpailuedun luomiseen perustuvat toimintatavat ovat usein alkuperäisen avoimuuden tavoitteen vastaisia.

4.2. Ohjelmistojen roolin muutos

Ohjelmistojen kehittäminen on muuttumassa suuntaan, jossa yritykset tasapainoilevat avoimen jakamisen hyötyjen kanssa ja kilpailuedun suojaamisen kanssa (Välimäki, 2005). Tiedonvaihdon asiantuntijoiden kesken arvellaan yleisesti lisäävän organisaation luovia kykyjä (esimerkiksi von Krogh ja von Hippel, 2006). Yhteisöperustainen kehittämistyö pystyy edelleen kohdistamaan uudelleen organisaatiossa väärällä tavalla kohdistettuja resursseja (Melian, 2001). Toisaalta kehitystyön tulokset haluttaisiin useimmiten salata kilpailijoilta niin kauan, että ne ehtivät tuottaa kaupalliselle organisaatiolle voittoa. Harva yritys olisi valmis maksamaan suuret kehittämiskustannukset ja sen jälkeen jakamaan kehittämistyön tulokset internetissä jokaiselle halukkaalle. Yksinkertaistaen voi siten väittää, että tasapainoilu avoimen ja suljetun alueen välillä vaikuttaa yrityksen valitsemaan yleiseen linjaan näitä kohtaan. Käytännössä tilanne on merkittävästi tätä monimutkaisempi: avoimen lähdekoodin käyttö yrityksissä tulee lähes aina tarkasteltavaksi yksittäistapauksena, koska ilmiön monimuotoisuus ei juuri mahdollista yleisten liiketoimintamallien kuvaamista tai jaottelua (Fink, 2003; Hecker, 1999).

Eurooppalainen ITEA-COSI projekti on tutkinut ohjelmistoalan muutosta ja kiteyttänyt sen kolmeen tekijään (ITEA-COSI, 2007). 1.) Muutos yrityksen sisäisestä kehittämisestä kohti alihankintaa ja integraattorin roolia, 2.) Muutos kohti liittoumia, ja 3.) Muutos kohti yhteistyötä avoimen lähdekoodin kehittäjäyhteisöjen kanssa. Kaikki kolme muutosta tarjoavat avoimen lähdekoodin hyödyntämiselle organisaatioissa lisämahdollisuuksia tulevaisuudessa.

4.3. Avoin lähdekoodi yritysten käytössä

Avointa lähdekoodia voidaan hyödyntää karkeimmalla tasolla ulospäin suuntautuvaan kehittämiseen (outbound OSS, Fink 2003) ja sisäänpäin suuntautuneeseen kehittämiseen (inbound OSS, Fink 2003). Näiden ero on siinä, että ulospäin suuntautuneessa lähestymistavassa viedään jotain organisaatiosta avoimelle alueelle ja Sisäänpäin suuntautuneessa siirretään jotain avoimelta alueelta organisaation sisälle. Ulossuuntautuneessa kehittämisessä yrityksen panos suuntautuu avoimelle alueelle (open domain) kuten internetiin. Sisäänpäin suuntautuneessa kehittämistyössä yrityksen panos suuntautuu vain yrityksen sisälle (closed domain), sen omaan käyttöön, kuten intranetiin. Ongelmakentät ovat näissä kahdessa hyödyntämistarkoituksessa melko erilaiset, samoin prosessit, joiden avulla niitä hallitaan (Fink, 2003). Yleisesti ottaen suljetulle alueelle suunnatut panokset eivät altista yritystä riskeille samalla kuin julkiselle alueelle suunnatut panokset (Fink, 2003). Vaikuttaisi myös siltä, että kynnys käyttää avoimen lähdekoodin ohjelmistoja on huomattavasti alempi kuin niiden kehittäminen.

Eräs variantti hyödyntämistavoista on ns. Inner source eli ajatus siitä, että tarpeeksi suuren yrityksen sisälle luodaan avoimen lähdekoodin kaltainen kehittämisympäristö (Lindman, 2008). Lukuisat yritykset ovat käynnistäneet tällaisia yrityksen sisäisiä avoimen lähdekoodin projekteja ja palveluita. Esimerkkeinä tällaista yrityksistä voidaan mainita vaikkapa HP, Xerox, NSN ja Philips. Joissain tapauksissa tämän kehitysympäristön kehittämät ohjelmistot voidaan myöhemmin siirtää avoimelle alueelle.

Shared Source on Microsoftin (Matusow, 2005) näkemys ohjelmistojen jakamisen eduista suurasiakkaille. Shared source voidaan ymmärtää myös laajemmin (Riepula, 2008) ohjelmistojen lähdekoodilisenssien myyntinä tai yhteiskehittämisessä. Tällöin tuotekehityksen kulut saataisiin jaettua useammalle käyttävälle taholle kuitenkaan julkaisematta mahdollisesti kilpailuetua tuovaa ohjelmistoa avoimesti.

4.4. Avoin vai ei?

Voidaan esittää vakava huoli avoimen lähdekoodin tutkimuskohteen muutoksesta siirryttäessä tutkimuskohteen oikeuttamisesta sen hyödyntämiseen yrityksissä. Avoin ohjelmistokehitys on tullut lähtemättömästi yrityksiin. Samalla se on törmännyt organisaatioiden eri tasoilla perinteisimpiin ohjelmistokehityksen menetelmiin.

Tuloksena vaikuttaisi hyvin monissa tapauksissa olevan avoimuuden väheneminen. Tämä kehitys ei ole ollut poikkeuksetonta: monet, erityisesti pienemmät ohjelmistotalot ovat ottaneet avoimen kehityksen haasteet vastaan ja ajatelleet ohjelmistokehityksensä uudestaan avoimen lähdekoodin tarjoamien mallien valossa.

Toki voidaan esittää myös väite, että avoimuuden väheneminen ei ole mitenkään negatiivinen kehityskulku, vaan esimerkiksi luo työtä ja mahdollisuuksia suomalaisille ohjelmistoyrityksille. Monessa tapauksessa se kuitenkin myös kangistaa organisaatioita ja auttaa eniten yrityksiä, jotka ovat jo vahvoja markkinoilla.

Periaatteellisemmalla tasolla kysymys ohjelmistojen avoimuudesta on vieläkin monisyisempi. Erityisesti vapaiden ohjelmistojen puolestapuhujat ovat usein olleet ohjelmistojen kaupallistamista vastaan. Toisaalta avoin lähdekoodi lanseerattiin yrityshyväksynnän saavuttamiseksi ja käyttäjämäärien kasvattamiseksi hävittämällä vapaiden ohjelmistojen ”häviävä asenne”. Lanseerauksen yhteydessä ei niinkään pyritty muuttamaan ohjelmistojen kehitys- tai käyttötapoja. Kymmenen vuotta myöhemmin projekti vaikuttaa saavuttaneensa ainakin osan alkuperäisistä tavoitteistaan. Nyt tosin huomataan, että organisaatioiden ottaessa käyttöön avoimen lähdekoodin ilmiö on muuttunut, hyvään tai pahaan.

Mikäli kävisi niin, että avoin lähdekoodi menettäisi avoimuutensa, niin siinä vaiheessa voidaan aiheellisesti kysyä, onko se enää edes käsitteellistä jatkumoa vaikkapa vapaiden ohjelmistojen idealle? Nähdäkseni avoimuuden menetys, vaikkakin tuottaisi lyhyellä aikavälillä merkittävää taloudellista hyötyä, johtaa silti pitkällä tähtäimellä ohjelmistoteollisuuden jäykistymiseen. Samalla saatamme menettää yhden lupaavimmista ohjelmistotuotannon tavoista. Toisaalta avoimen lähdekoodin lisensseillä on jo nyt julkaistu niin paljon ohjelmistoja, että ilmiö ei ole missään tapauksessa katoamassa mihinkään.

4.5. Tutkimusaiheita tulevaisuudessa

Avoimen lähdekoodin tutkimus voi tarjota edelleen paljon organisaatioille, jotka haluavat hyödyntää sitä toiminnassaan. Keskittyessään tapoihin, joilla organisaatiot sitä hyödyntävät, tutkimus palvelee myös laajemmin yhteiskuntaa. Avoimen lähdekoodin kehitystapojen ja käytön yleistyminen esimerkiksi julkisella sektorilla voisi hyödyttää merkittävästi kansantaloutta.

Kriittinen ja analyyttinen ote tutkimuskohteeseen on erityisesti avoimen lähdekoodin tutkimuskentällä tarpeen. Sangen usein nimittäin käy edelleen niin, että itse ilmiö, sen syyt ja seuraukset, hukkuvat joko liikaan innostukseen tai sitten perusteettomaan pessimismiin. Tilannetta ei helpota se, että avoimen lähdekoodin käsitettä käytetään tieteellisen puheen lisäksi sangen löysästi joissain muissa yhteyksissä. Tähän taas on osasyynä se, että avointa lähdekoodia pidetään homogeenisenä ilmiönä: ihan kuin kaikki yritykset ja kehittäjäyhteisöt olisivat samankaltaisia. Jo pinnallinen tutustuminen aiheeseen saa kyseenalaistamaan tämän oletuksen: pikemminkin hämmentävää on ilmiön monimuotoisuus. Monimuotoisuus johtaa hyvin nopeasti vaikeuksiin, kun yritetään selvittää analyyttisesti mistä ilmiössä on oikein edes kysymys - ja miten tätä tietoa voidaan hyödyntää vaikkapa organisaation kehittämisessä.

Eräs mielenkiintoisimmista aiheista ovat erilaiset avoimen ja suljetun (open and closed domain) väliin jäävät hybridit. Miten ne toimivat ja mitä niillä saavutetaan? Miten käy vaikkapa innovaatioiden? Miten tapahtuu empiirisesti kun kehittäjäyhteisöt törmäävät tai elävät yhdessä hierarkkisempien organisaatioiden kanssa?

Tutkimuskentällä keskeisenä tavoitteena tulisi pitää avoimuuden säilyttämistä mukana keskustelussa. Tai jos siitä luovutaan, niin silloin pitää luopua myös termistä avoin lähdekoodi.

Lähteet

Cusumano, M. (2004). The Business of Software. Free Press. New York.

COSI. http://www.itea-cosi.org

Fink, M. (2003). The Business and Economics of Linux and Open Source. Prentice Hall, New Jersey.

Fitzgerald, B. (2006). "The Transformation of Open Source Software". MIS Quarterly, 30 (4), 587-598.

Hecker, F. Setting Up Shop: The Business of Open-Source Software. IEEE Soft-

ware, 16(1):45{51, January-February 1999.

Matusow, J. (2005) Shared Source: Microsoft Perspective in Feller, J&Fitzgerald, B. Perspectives on Free and Open Source Software.

Riepula, M., Lindman, J. (2008). Determinants of commercial source code licensing – going beyond the obvious answer of don't. (Julkaisematon työpaperi).

Melian, C.,. Ammirati, CB ., Garg, P., and Sevon, G. Building Networks of Software Communities in a Large Corporation. Hewlet Packard POS, 2001.

von Krogh, G., von Hippel, E. (2006). The Promise of Research on Open Source Software. Management Science, 52(7):975-983.

Välimäki, M. (2005). *The Rise of Open Source Licensing. A Challenge to the Use of Intellectual Property in the Software Industry*, Helsinki University of Technology, Helsinki.

5. Avoimen lähdekoodin kehittäjäyhteisöt sosiaalisena ilmiönä

Teemu Mikkonen
Tutkija, Hypermedialaboratorio,
Tampereen yliopisto
teemu.mikkonen@uta.fi

Abstrakti

Tässä kirjoituksessa tarkastelen avoimen lähdekoodin yhteisöjä kolmen Emile Durkheimiltä soveltamani käsitteen kautta. Ensimmäinen näistä on ”hypersiviliaatio”, jonka näen edellytyksenä yhteisöjen synnylle ja siksi olennaisena puhuttaessa tämän tapaisista yhteisöistä. Toinen käsite on ”tiedollinen solidaarisuus”, joka on lisänä kahdelle Durkheimin solidaarisuuden lajille (altruistinen ja egoistinen). Tiedollinen solidaarisuus liittyy kolmanteen käsitteeseen, ”anomia”, jonka näen ehdottomana edellytyksenä yhteisöjen olemassaololle ja kehitykselle. Toisin sanoen esitän, että tiedon ”vapaa” jakaminen ja jaettavuus, sekä yhteiset, mutta joustavat tulkintakehykset ovat omiaan pitämään yhteisöt kehittyvinä ja elinvoimaisina.

5.1. Johdanto

> ”Yhteiskunnassa ei ole muita aktiivisia voimia kuin yksilöt, mutta silti nämä yksilöt muodostavat yhdistymällä uudenlaisen psyykkisen olion, jolla niinmuodoin on oma tapansa ajatella ja tuntea.” (Durkheim 1985, 379)

Internetin synty ja tietotekniikan kehittyminen ovat luoneet uusia globaaleja yhteisyyksiä. Tietotekniikan puolella yksi esimerkki näistä globaaleista yhteisöistä on avoimen/vapaan lähdekoodin ohjelmistojen (Free/Libre Open Source Software, FLOSS) kehittäjäyhteisöt. Ne muodostavat suuren ja hajanaisen yhteisön, joka on keskittynyt erilaisten tietoteknisten ohjelmistojen ja niihin liittyvien oheistuotteiden kehittämiseen, käyttämiseen ja arvioimiseen. Kehittäjäyhteisöt poikkeavat monista muista yhteisöistä niiden hajanaisuuden ja erilaisten lähtökohtien vuoksi (ks. Mikkonen, Vaden & Vainio 2007). Siksi niiden määrittely yhteisöinä, vaatii omanlaistaan kysymyksen asettelua. Koska sosiaalisena ilmiönä avoimen/vapaan lähdekoodin yhteisöt ovat monimuotoinen ja

erityinen ilmiö, asetankin kysymyksen seuraavasti: ”Millaisia sosiaalisia erotteluja tarvitaan, jotta yhteisöjä voidaan tutkia?” Tässä kirjoituksessa tuon esille muutamia erotteluja, jotka mielestäni sopivat erityisesti avoimen/vapaan lähdekoodin kehittäjäyhteisöjen tutkimiseen.

5.2. Hypersivilisaatio ja avoimen lähdekoodin kehittäjäyhteisöt

Emile Durkheimin (1990) mukaan nykyinen yhteiskunta kehittyi yhteiskunnallisen työnjaon kasvaessa. Tätä yhteiskunnallisen työnjaon kehittymistä hän kuvaa siirtymisellä "mekaanisesta solidaarisuudesta" "orgaaniseen solidaarisuuteen". Mekaaninen solidaarisuus viittaa maalaisyhteisöihin tai kyläyhteisöihin, joissa yhteisön solidaarisuus ja koossapysyminen on riippuvainen jäsenten "samanlaisuudesta". Luonteenomaista näille yhteisöille on, että erilaisuudesta on säädetty ankarat rangaistukset ja normit perustuivat pääosin vanhoihin traditioihin. Orgaaninen solidaarisuus sitä vastoin on käsite jolla Durkheim kuvaa yhteiskunnan työnjaon eriytymisen kautta ”syntynyttä” individualismia ja yhteisten arvojen pirstaloitumista. Kun työnjako yhteisöissä kasvaa, ”samuus” heikkenee ja solidaarisuus muuttuu organisoidummaksi. Tässä muutoksessa ns. kollektiivinen tajunta (jolla Durkheim viittaa yhteisiin arvoihin) ja yhteisön yhteenkuuluvuuden tunne heikkenee. (Durkheim 1990, Kivisto 2004.)

Siirtyminen mekaanisesta solidaarisuudesta orgaaniseen solidaarisuuteen on lähellä sitä, mitä kuvataan usein traditionaalisesta yhteiskunnasta moderniin yhteiskuntaan siirtymisellä (Kivisto 2004, 122-123). Durkheim lähestyy positiivisessa mielessä sitä solidaarisuuden tapaa mitä on usein kuvattu postmoderniksi kirjoittaessaan ns. hypersivilisaatiosta (Baumann 1999, Kivisto 2004, 122-123; Sulkunen 1999, 287-289). Kun solidaarisuus yhteiskunnassa muuttuu yhä orgaanisemmaksi, muuttuu myös yksilöiden tapa toimia erilaiseksi. Yhteisöjen siteiden heiketessä tulee yksilöille mahdollisuus vaihtaa yhteisöä omien taipumustensa mukaan. Durkheim viittaa tähän seuraavasti.

> ”Hypersivilisaatio, joka synnyttää anomistista ja egoistista taipumusta, myös jalostaa hermojärjestelmiä tehden niistä äärimmäisen valikoivia.” (Durkheim, 1985, 396)

Tietoyhteiskunnan voisi mitä suurimmassa määrin nähdä olevan kuvatunlainen ”hypersivilisaatio”. Internet on mahdollistanut sen, että yksilöiden ei tarvitse enää rajoittua paikallisiin tai oman kulttuuripiirin yhteisöihin ja niiden normijärjestelmiin, vaan hän voi valita monista eri mahdollisuuksista omaa yksilöllisyyttään tukevan yhteisön. Toisaalta yksilöllisyydestä

itsestään on tullut normi. Kun on tullut mahdollisuus valintaan, yksilölle avautuu yhä enemmän velvoitteita tehdä valintoja yhä laajemmasta määrästä vaihtoehtoja. Durkheim (1985) puhuu tässä yhteydessä "ihmisyyden uskonnosta" tai "persoonallisuuden kultista", jolla hän viittaa siihen, että yksilöllisyydestä itsestään on tullut pyhä. Se, että yksilö saa ja joutuu itse päättämään omista valinnoistaan, joiden tulisi mahdollisimman paljon heijastaa hänen persoonallisuuttaan, on uskonnon kaltainen työnjaon tuloksena syntynyt periaate. Tähän periaatteeseen viittaa myös avoimen/vapaan lähdekoodin -kehittäjäyhteisöjen takana oleva filosofia (Stallman 1999).

> If I am not for myself, who will be for me?
>
> If I am only for myself, what am I?
>
> If not now, when?

Näin Richard Stallman (1999) perustelee syitä ja henkeä aloittaa GNU – projekti Hilleliltä lainaamallaan sitaatilla. Tavoitteena hänellä oli yhdistää itsensä toteuttaminen yhdessä tekemiseen ja yhteisöön tärkeimpänä periaatteenaan vapaus (jolla hän haluaa myös korostaa eroja avoimen lähdekoodin tuotteisiin). Vapaudella hän viittaa esimerkiksi siihen, että jokaisella on mahdollisuus muokata ohjelmaa vastaamaan omia tarpeitaan. Erityisenä Stallmanin esittelemässä individualismissa on, että siinä vapaus ja avoimuus yhdistyvät yhteiseen tekemiseen.

Kehittäjäyhteisöissä tärkeänä periaatteena on siis individualismin ja valinnan mahdollisuuden periaate. Tämä periaate on ollut kehittäjäyhteisöjen vapaaehtoisen muodostumisen ja toiminnan takana, ylläpitäjänä ja joskus myös hajottajana. Yksi esimerkki siitä kuinka valinnan vapaus on jakanut yhteisöitä, on edelleen jatkuva kiista avoimen ja vapaan lähdekoodin, sekä avoimen ja osittain avoimen lähdekoodin kehittäjäyhteisöjen kesken. Näihin kiistoihin liittyy se, millaisia motiiveja yhteisöjen jäsenillä on. Yhteisöllisten motiivien painottuneisuus erottaa vapaat, avoimet ja ”puoliavoimet” yhteisöt jonkin verran toisistaan (ks. Lindmanin edellinen artikkeli), mutta niitä kaikkia yhdistää ns. ”tiedollinen solidaarisuus”.

5.3. Kehittäjäyhteisöt ja tiedollinen solidaarisuus

Yhteisöllisyyden aste on yhteydessä siihen miten ”kollektiivisia patologisia ongelmia” esiintyy. Liian sitova yhteisö (altruismi) pysäyttää kehityksen, koska erilaisuudella ja vapaudella ei ole tarpeeksi tilaa aiheuttaa muutosta. Liika yksilöllisyys (egoismi) taas sitä vastoin erottaa yksilöt liian peruuttamattomasti yhteisöstä ja näin estää kehityksen. (Durkheim 1985.)

Kehittäjäyhteisöissä yhteisöllisyyden astetta on hankala varsinaisesti mitata, mutta sitä voi teoreettisella tasolla spekuloida. Yksi tapa erotella ”yhteisöllisemmät” ja yksilöllisemmät on erotella yhteisöt sen mukaan, mitä kehittäjät ilmoittavat motiiveikseen. Luokittelemalla motiivit ”itseä hyödyttäviksi” (self-enriching) ja ”ryhmää hyödyttäviksi” (group-enriching) voidaan vertailla ryhmien jäsenten motiivien erilaisia painotuksia (Aalbers 2004; Mikkonen, Vaden & Vainio 2007). Ne ryhmät, joissa kehittäjien motiivit ovat pääosin luokiteltavissa itseä hyödyttäviksi, voi nähdä egoistisina yhteisöinä ja päinvastoin ne joissa painottuu ryhmää hyödyttävät motiivit, altruistisina. Tästä vastakohtaparista näyttäisi yhteisöissä painottuvan yhteisölliset motiivit jonkin verran enemmän kuin yksilölliset. Tosin nämä painotukset ovat paljolti yhteisöriippuvaisia ja erityisesti palkkaperusteisemmissa yhteisöissä egoistisemmat motiivit ovat hiukan painottuneempia. (Mikkonen, Vaden & Vainio 2007.)

Kehittäjäyhteisöjen kohdalla voisi edellä esitettyjen motiivien (tai solidaarisuuden) lisäksi erottaa ns. tietoa kartoittavat motiivit. Tieto tässä viittaa tietoon ja tiedon kehitykseen yleensä, vaikka tiedon objektiivisesta lisääntymisestä ja kehityksestä ei välttämättä ole yksimielisyyttä. (Mikkonen, Vaden & Vainio 2007.) Tässä sillä tarkoitetaankin tiedon lisääntymiseen viittaavaa puhetapaa.

Avoimen lähdekoodin -kehittäjäyhteisöissä tiedon lisääminen näyttäisi olevan merkittävä motiivi perustella vapaaehtoistyön tekemistä. Olen hahmotellut motiivien perusteella eräänlaisen solidaarisuuspyramidin (vrt. Maslovin tarvehierarkia, Maslow 1943), jota voidaan kehittäjäyhteisöjen lisäksi mahdollisesti soveltaa myös muihin "tietoa kartuttaviin" vapaaehtoisyhteisöihin (esim. Wikipedia). Kuviossa motiivit on aseteltu pyramidin muotoon siinä järjestyksessä, missä kehittäjät käyttivät niitä perusteinaan. (ks. Table 4., Mikkonen, Vaden & Vainio 2007). Voidaan ajatella, että itseä hyödyttävät motiivit ovat se pohja, jolla harvemmin perustellaan vapaaehtoistyötä, mutta joka kuitenkin on välttämätön "ylemmille" motiiveille. Tietoa kartoittavat motiivit sitä vastoin ovat suosituin perustelu vapaaehtoistyölle, mutta se tarvitsee kuitenkin "pohjakseen" ryhmää hyödyttävät ja itse hyödyttävät motiivit. Ideana tässä kuviossa on, että ylempi taso on aina riippuvainen alemmasta (ei välttämättä kaikista yksittäisistä väitteistä), mutta se ei silti ole alemman tason summa (Durkheim 1985). Pyramidin voi nähdä sisällöllisesti versiona Maslowin (1943) tarvehierarkiasta, mutta se perustuu kuitenkin avoimen lähdekoodin kehittäjäyhteisöistä tehtyyn kansainväliseen kyselyyn ja siitä tehtyyn faktorianalyysiin (Mikkonen, Vainio & Vaden 2006).

Kuva 1. Solidaarisuuspyramidi

Durkheim (1985) itse ei puhu tiedon lisääntymisestä omana sosiaalisena ulottuvuutenaan. Se vaikuttaisi kuitenkin olevan monissa yhteisöissä merkittävä yhteisöä koossapitävä voima. Voidaankin miettiä onko Durkheimin esille tuomien yhteisöllisen solidaarisuuden ja egoistisen solidaarisuuden rinnalla oma itsenäinen ulottuvuus, tiedollinen solidaarisuus.

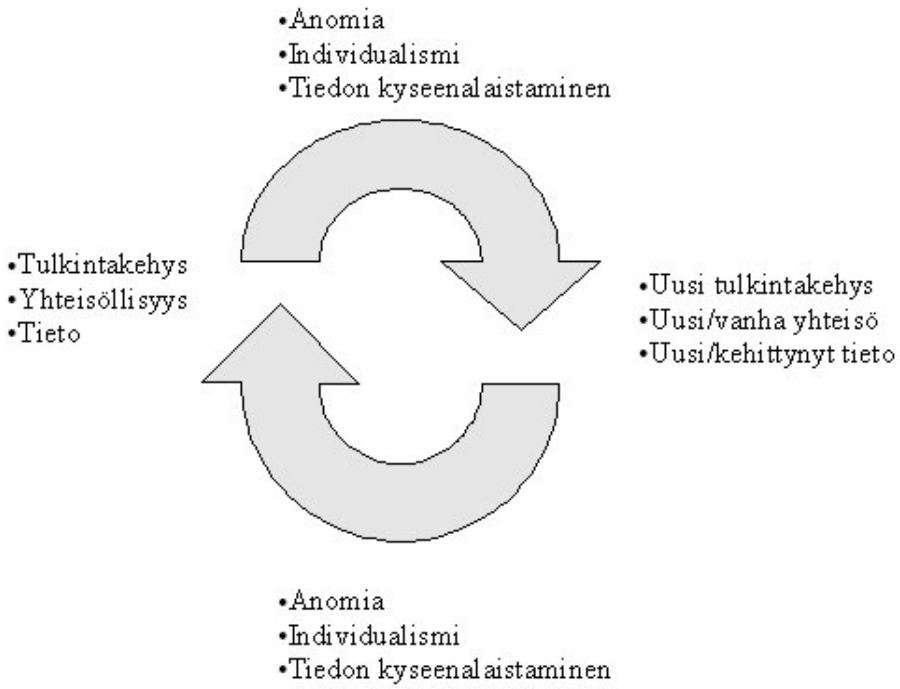
5.4. Anomia yhteisöissä

> "Ihmiskunnan historia on juuri ajatuksen vapauden edistyksen historiaa."
>
> (Durkheim 1985, 467)

Anomian Durkheim liittää ”avoimuuteen edistyksen ajatuksille” (Durkheim 1985, 393). Liika anomia aiheuttaa patologisia tiloja, mutta sopivissa määrin se on edellytys yhteiskunnan edistymiselle. Anomian voisi nähdä olevan eräänlaisena avoimien kehittäjäyhteisöjen liikkeellepanijana. Internetin

tarjoama paikallinen ”rajoittamattomuus”, sekä kehittäjien taitojen mahdollistama tietotekninen epänormatiivisuus on siinä purkautunut kansainväliseksi yhteisöllisyydeksi. Näissä yhteisöissä on muodostunut uusia säännöksiä ja normatiivisuuksia. Nämä säännöt ja normit ovat vapaaehtoisia yhteisön jäsenille ja täten sallivat myös epänormatiivisten kehittelyiden tulon teknisen "diskurssin" piiriin. Toisin sanoen sääntöjen olemassaolo luo edellytykset yhteisön ja yhteisen puhetavan syntymiselle, mutta niiden hallitsematon rikkominen avaa tietä sääntöjen testaamiselle ja kehittymiselle. Ilman jonkinlaista yhteistä tulkintakehystä yhteisö ei pystyisi tehokkaasti toimimaan, mutta toisaalta liian rajoittava tulkintakehys on este sen kehitykselle (tai sen tuloksena tulleelle "kehitykselle). Normatiivisuuden aste onkin yhteydessä yhteisön tehokkuuteen ja koossapysymiseen. Seuraavassa kuviossa on hahmoteltu sitä, miten anomia, individualismi ja tiedon kyseenalaistaminen toimivat olemassa olevien ja uusien tulkintakehysten, sekä yhteisöjen ja tiedon välittäjinä. (vrt. Kuhn 1995, Heritage 1996.)

Kuva 2. Anomiakehä



Uudella tulkintakehyksellä ei tässä tarkoiteta välttämättä uutta siinä mielessä, että se olisi välttämättä "kehittyneempi" tulkintakehys, vaan paremminkin, että se on erilainen (vrt. paradigma, Kuhn 1995). Yhteisö saattaa olla sama tai eri kun tulkintakehys ja tieto on muuttunut, mutta täydellinen muutos saattaa myös johtaa edellisen yhteisön hajoamiseen ja uuden syntymiseen. Tästä esimerkkinä ovat monet loppuneet tai

lopetetut projektit, joiden "raunioille" on syntynyt uusia projekteja. Tuotteen kehittymisessä, vanhan hyväksikäytössä ja uusien tarkoituksenmukaisten projektien synnyssä on avoimen lähdekoodin kehittäjäyhteisöjen vahvuus verrattuna suljettuihin kehittäjäyhteisöihin se, että vaikka yksittäiset kehitysprojektit hiipuisivat voi periaatteessa kuka vaan jatkaa siitä mihin niissä on jääty. Oikeanasteinen anomia, jota avoin kehitysmalli edesauttaa, antaakin mahdollisuuden uusien "innovaatioiden" syntymiselle huomattavasti suljettua mallia tehokkaammin.

5.5. Yhteenvetoa ja johtopäätöksiä

Valinnan mahdollisuus/pakollisuus on kulkenut käsi kädessä tiedon lisääntymisen ja säännöistä vapautumisen kanssa. Yhteiskuntarakenteiden, uskonnon ja monen muun käsitejärjestelmän tiedollisten rajoitusten hellittäessä on myös solidaarisuus muuttanut muotoaan. Yhteisöllisyys ei välttämättä ole kadonnut, vaikka individualismi olisi saanut lisää jalansijaa, se on vain saanut uuden, globaalin viitekehyksen. Tätä on mahdollistanut ja tämä on mahdollistanut se(n), että tietotekniikan välityksellä on syntynyt yhteisöjä, jotka perustuvat ns. tiedolliselle solidaarisuudelle. Avoimen/vapaan lähdekoodin -kehittäjäyhteisöjen erilaisuudesta ja moninaisuudesta huolimatta yhdistää niitä kaikkia se, että kehitettävien ohjelmien tulisi tulla olemaan mahdollisimman toimivia. Informaation voisikin nähdä näiden yhteisöjen osalta ottaneen osansa siitä "tavarafetissin" roolista, joka ennen on liitetty rahaan (Sulkunen 1999).

Yhteisöjen toiminnan jatkuvuuden ja niistä hyötymisen kannalta olennaista on tiedostaa se, että mikäli avoimet informaatiovirrat ja kehitys pysähtyy, usein myös yhteisö hajoaa tai muuttaa muotoaan. Tämän takia on yhteisön jatkuvuuden osalta tärkeää ensinnäkin se, että on jotain kehityttävää ja toiseksi se, että kehitys on tehty mahdolliseksi. Kuviossa 2. yritin havainnollistaa sitä kehää, joka syntyy jatkuvasta vanhan tulkintakehyksen, anomian ja uuden tulkintakehyksen vaihtelusta. Näkemykseni mukaan ilman tätä anomian vaihetta yhteisö ei pysty kehittymään ja tätä kautta se joko sulautuu toiseen projektiin tai kuihtuu pois. Täten siis esimerkiksi se, että lähdekoodin näkyvyyttä yritetään rajoittaa sopimuksin tai että ”avoimia” kehittäjäyhteisöjä yritetään luoda yritysten sisään (ks. Lindemanin artikkeli edellä) voi olla ensimmäinen este kehitykselle ja lopulta hajottaa koko yhteisön tuotteen ympäriltä. Kehitys vaatii läpinäkyvyyttä ja vapaasti organisoitunutta yhteisöä ympärilleen. Ainakin jos sitä kutsutaan avoimeksi.

> ”Mutta ainoat elinkelpoiset ovat ne, jotka sallivat suuremman tutkimisen vapauden ja yksilöllisen aloitteisuuden…” (Durkheim 1985, 467)

Lähteet

Aalbers,Martine (2004): Motivation for participation in an open source software community. Saatavissa: <http://download.blender.org/documentation/bc2004/Marhttp://www.firstmonday.org/issues/issue12_2/mikkonen/index.htmltine_Aalbers/results-summary.pdf>, (luettu 5.2.2008).

Baumann, Zygmunt (1999) Sosiologinen ajattelu. Vastapaino, Tampere.

Durkheim, Emil (1985): Itsemurha. Tammi, Helsinki. Ranskankielinen alkuteos julkaistu 1897.

Durkheim, Emil (1990): Sosiaalisesta työnjaosta. Gaudeamus, Helsinki. Ranskankielinen alkuteos julkaistu 1893.

Heritage, John (1996): Harold Garfinkel ja etnometodologia. Gaudeamus, Jyväskylä.

Kivisto, Peter (2004): Key Ideas in Sociology. Pine Forge Press, London.

Kuhn, Thomas (1995): Tieteellisten vallankumousten rakenne. Suomentanut Kimmo Pietiläinen. Art House, Helsinki.

Maslov, Abraham (1943): A Theory of Human Motivation. Psychological Review, 50, 370-396.

Mikkonen, Teemu, Vadén Tere & Vainio, Niklas (2007): The Protestant ethic strikes back: Open source developers and the ethic of capitalism. First Monday -verkkojulkaisu. Saatavissa: <http://www.firstmonday.org/issues/issue12_2/mikkonen/index.html>(luetttu 5.2.2008).

Mikkonen, Teemu, Vainio Niklas & Vadén Tere (2006): Survey on four OSS communities: description, analysis and typology. e-Business Research Center Research Reports 33. Tampere University of Technology & University of Technology. Saatavissa: <http://www.coss.fi/web/coss/research/ossi/publications> (luettu 5.2.2008).

Simmel, Georg (2005) Suurkaupunki ja moderni elämä. Gaudeamus, Helsinki

Stallman, Richard (1999): The GNU Operating System and the Free Software Movement. Kirjassa "Open Sources : Voices from the Open Source Revolution". O'Reilly. Saatavissa: <http://www.oreilly.com/catalog/opensources/> (luettu 5.2.2008).

Sulkunen, Pekka (1999) Johdatus sosiologian. - Käsitteitä ja näkökulmia. WSOY, Helsinki.