

Stud.techn. Atle Frenvik Sveen

Use of Free and Open Source GIS in Commercial Firms

Trondheim, December 18, 2008



Project title: Use of Free and Open Source GIS in Commercial Firms	Date: December 18, 2008		
	Pages (inc. appendix): 118		
	Master's Thesis	Project assignment	X
Name: Stud.techn. Atle Frenvik Sveen			
Teaching supervisor: Terje Midtbø			
External technical contacts/supervisors: n/a			

Abstract:

This study examines the use of FOS GIS in a commercial setting, and the objective is twofold; to provide a method for selecting a FOS GIS and to give an overview of the status of a subset of the available FOS GIS products, desktop GISes. The main GIS features are management, manipulation and analysis of spatial data. A desktop GIS differs from a Web/Internet GIS in the sense that it is installed on a PC, allowing more complex tasks to be carried out.

Free and Open Source Software (FOSS) is software released under a license ensuring the users the freedom to use, modify and redistribute the software. Access to the source code is a prerequisite for this. The later years have brought a shift from the ideological focus to commercial adoption. Companies are attracted to FOSS due to the high quality and low price. Selection of FOSS products are typically carried out without the use of any formal evaluation method.

The FOS GIS community is centered around the OSGeo foundation, which serves as a host for several projects. Both commercial companies and governmental institutions, as well as community initiatives, initiate and develop FOS GIS projects. The existing research on FOS GIS is rather limited and tends to focus on well known products such as GRASS. None of the reviewed articles on FOS GIS indicated that frameworks for selecting FOSS products were used, neither did any describe any such methods. The findings showed that FOS GIS products in general are considered mature enough to compete with the proprietary solutions.

The use of a formal method for assessing FOS GIS in a commercial setting ensures that a better choice will be made, as more candidates will be examined and the risk of evaluator prejudice is minimized. Three existing frameworks for evaluation of FOSS products are examined. The general method consists of four phases; an initial gathering of candidates, gathering of measurable data, definition of evaluation criteria and a final evaluation. The risk of using a framework that only uses scores and no human assessment is discussed, and the need for human judgment emphasized. A tailored method is constructed based on the findings from the frameworks and a study on FOSS adoption.

An example evaluation is carried out, using a real life case. The steps are covered in detail in order to serve as an example. Methods for gathering required information and deciding what criteria to use are discussed. The evaluation is concluded and the need for the utilization of test installations is emphasized. The probability that that more FOS GIS projects will emerge emphasizes the need for a method for selecting the right FOS GIS.

Keywords:

1. Free and Open Source GIS (FOS GIS)
2. FOS GIS evaluation
3. uDig, gvSIG, OpenJUMP
4.

Abstract

This study examines the use of FOS GIS in a commercial setting, and the objective is twofold; to provide a method for selecting a FOS GIS and to give an overview of the status of a subset of the available FOS GIS products, *desktop GISes*. The main GIS features are management, manipulation and analysis of spatial data. A desktop GIS differs from a Web/Internet GIS in the sense that it is installed on a PC, allowing more complex tasks to be carried out.

Free and Open Source Software (FOSS) is software released under a license ensuring the users the freedom to use, modify and redistribute the software. Access to the *source code* is a prerequisite for this. The later years have brought a shift from the ideological focus to commercial adoption. Companies are attracted to FOSS due to the *high quality* and *low price*. Selection of FOSS products are typically carried out without the use of any formal evaluation method.

The FOS GIS community is centered around the OSGeo foundation, which serves as a host for several projects. Both commercial companies and governmental institutions, as well as community initiatives, initiate and develop FOS GIS projects. The existing research on FOS GIS is rather limited and tends to focus on well known products such as GRASS. None of the reviewed articles on FOS GIS indicated that frameworks for selecting FOSS products were used, neither did any describe any such methods. The findings showed that FOS GIS products in general are considered *mature enough* to compete with the proprietary solutions.

The use of a formal method for assessing FOS GIS in a commercial setting ensures that a better choice will be made, as more candidates will be examined and the risk of evaluator prejudice is minimized. Three *existing frameworks* for evaluation of FOSS products are examined. The general method consists of four phases; an initial gathering of candidates, gathering of measurable data, definition of evaluation criteria and a final evaluation. The risk of using a framework that only uses scores and no human assessment is discussed, and the need for *human judgment* emphasized. A *tailored method* is constructed based on the findings from the frameworks and a study on FOSS adoption.

An example evaluation is carried out, using a real life case. The steps are covered in detail in order to serve as an example. Methods for gathering required information and deciding what criteria to use are discussed. The evaluation is concluded and the need for the utilization of test installations is emphasized. The probability that more FOS GIS projects will emerge emphasizes the need for a method for selecting the right FOS GIS.

Preface

This study was carried out as part of the course TBA4560 at the Division of Geomatics, NTNU, Norway. The subject of the study was refined by Terje Midtbø at NTNU. This work has made me realize that FOSS products are usable in a commercial setting and I have learned a lot, both with regard to the GIS history and the FOSS movement.

As I wrote in an email to one of the gvSIG developers:

“One of the reasons FOSS makes an interesting research subject is the fact that the communities takes time to answer my questions and are generally cooperative.”

I would like to thank the following FOS GIS developers for answering my inquiries regarding their projects; Jesse Eichar (uDig), Victoria Agazzi and César Martínez Izquierdo (gvSIG), and Stefan Steiniger and “Sunburned Surveyor” (OpenJUMP). Their replies, clarifications and help have made my evaluation easier.

I would also like to thank the company serving as a case in this project (who wish to remain anonymous) for participating in the interview and answering my follow-up questions. I hope that this study will make their considered FOSS adoption easier.

Last, but not least, I would like to thank my dear Birgitte for supporting me throughout the work on this project. I love you!

Trondheim, December 18, 2008

Atle Frenvik Sveen
atle@frenviksveen.net

License

This version of the report is licensed under the *Creative Commons Attribution-NoDerivs 3.0 Unported* License. In short, this means that you are free:

- to Share – to copy, distribute and transmit the work

Under the following conditions:

- Attribution. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- No Derivative Works. You may not alter, transform, or build upon this work.

For the full license text, see <http://creativecommons.org/licenses/by-nd/3.0/legalcode>.

In order to attribute the work, you may simply refer to my name (Atle Frenvik Sveen) and provide a link to my webpage (<http://atlefren.net>). Should my webpage be shut down in the future the mentioning of my name will be sufficient attribution.

Contents

1	Introduction	1
2	What is a Desktop GIS?	3
2.1	GIS - Geographic Information Systems	3
2.2	Web/Internet GIS	5
2.3	Desktop GIS	6
3	What is Free and Open Source Software?	9
3.1	Free Software and Open Source Software	9
3.2	Licenses	11
3.3	Why use FOSS?	12
4	FOS GIS Overview	15
4.1	Organizations	15
4.2	Sources and Sponsors	17
4.3	FOS GIS Research	18
5	A Method for Evaluating FOS GIS	23
5.1	Why use a Framework?	23
5.2	Existing frameworks and Literature on FOSS Evaluation	24
5.3	A Tailored Method	27
6	Case Study	31
7	Examination and Evaluation of Candidate Projects	33
7.1	Usage Scenarios and Requirements	33
7.2	Initial Filtering	40
7.3	Data Collection	43
7.4	Ranking and Selection	56
8	Recommendations	61
8.1	Analysis	61
8.2	Final Recommendation	62
9	Further Work	63
9.1	Test Installation Guidelines	64
9.2	Usability Test Guidelines	65

10 Discussions and Conclusion	67
10.1 Limitations - Discussion	67
10.2 Data Availability - Discussion	68
10.3 Final Selection - Discussion	69
10.4 Concluding Remarks	69
References	71
Appendix	A-1
A Questions Used in the Interview	A-1
B Evaluation Criteria, Tables	A-3
C Unique Evaluation Criteria	A-13
D Identity Card and Evaluation Sheet Templates	A-15
E Calculation of Requirement Scores	A-17
F FOS GIS History Links	A-19
G Links to Project Websites	A-21
H Calculation of Code Statistics	A-23
I Mailing List Statistics	A-25

List of Tables

3.1	The three FOSS license categories	11
4.1	Examples of OSGeo hosted projects	16
4.2	Different FOS GIS products ordered by source.	18
6.1	FOSS usage scenarios for the analyzed system.	31
7.1	Grouped requirements for the wanted FOS GIS	34
7.2	Required functionality for the system	36
7.3	Final requirements for the wanted FOS GIS	40
7.4	Candidate FOS GIS projects	41
7.5	Functionality of the candidates	42
7.6	uDig Identity Card	46
7.7	uDig Evaluation Sheet	47
7.8	gvSIG Identity Card	50
7.9	gvSIG Evaluation Sheet	51
7.10	OpenJUMP Identity Card	54
7.11	OpenJUMP Evaluation Sheet	55
7.12	Evaluation results	56
B.1	Criteria for evaluation of required functionality.	A-3
B.2	Criteria for evaluation of evolution direction.	A-3
B.3	Criteria for evaluation of target platform compliance.	A-3
B.4	Criteria for evaluation of reliability	A-4
B.5	Criteria for evaluation of maintainability	A-5
B.6	Criteria for evaluation of community activity	A-6
B.7	Criteria for evaluation of support availability	A-7
B.8	Criteria for evaluation of project longlivity	A-8
B.9	Criteria for evaluation of flexible maintenance	A-8
B.10	Criteria for evaluation of cost reduction potential.	A-9
B.11	Criteria for evaluation of possibility to influence further development	A-10
B.12	Criteria for evaluation of possibility to decrease proprietary dep. . .	A-11
D.1	Identity Card template	A-15
D.2	Evaluation Sheet template.	A-16

List of Figures

2.1	A comparison of typical setups for Web/Internet and Desktop GIS.	7
3.1	Free Software and Open Source Software illustrated	10
4.1	The OpenGIS stack	22
5.1	Phases of the BRR and QSOS frameworks	27
5.2	Evaluation method for FOS GIS illustrated	29
7.1	Graph showing the scores presented in table 7.12.	57

List of Abbreviations and Definitions

ArcSDE – A spatial database extension from ESRI

BSD License – Berkeley Software Distribution License

COTS – Commercial off-the-shelf software

ESRI – The leading proprietary GIS vendor

FOS GIS – Free and Open Source GIS

FOSS – Free and Open Source Software

FSF – Free Software Foundation

GIS – Geographic Information System

GML – Geography Markup Language

GNU/Linux – FOSS Operating System

GPL – GNU Public License

GUI – Graphical User Interface

MPL – Mozilla Public License

OGC – Open Geospatial Consortium

OGR – A FOSS library providing read (and write) access to a variety of vector file formats (OSGeo, 2008)

OSGeo – The Open Source Geospatial Foundation

OSI – Open Source Initiative

SLD – Styled Layer Descriptor (OGC, 2002b)

SOSI – Samordnet Opplegg for Stedfestet Informasjon

SVN – Subversion, a Revision or Version Control System

WFS – Web Feature Service

WMS – Web Map Service

1 Introduction

“The Open Source GIS space includes products to fill every level of the OpenGIS spatial data infrastructure stack. Existing products are now entering a phase of rapid refinement and enhancement, using the core software structures that are already in place. Open Source software can provide a feature-complete alternative to proprietary software in most system designs.”

These words formed the introduction to a survey of Open Source GIS given on the FOSS4G Conference¹ in 2007 (Ramsey, 2007). They summarize the motivation for this project assignment in an elegant manner. In the later years Free and Open Source Software (FOSS) has been recognized as a real alternative to proprietary solutions, and several commercial firms and governmental institutions are adopting FOSS products in order to decrease license fees, to get access to high quality software or to become independent of proprietary vendors (OpenBRR, 2005).

This trend also applies to the GIS market, where FOSS solutions now cover most software needs and are capable of delivering high quality software. The recent advances in on-line mapping sites and GIS solutions have broadened the scope of the traditional Geographic Information Systems, and there is an increase in software developers developing such solutions. Several of these developers and companies choose to release their work under a FOSS compatible license.

This study aims to provide a method for evaluating such Free and Open Source GIS projects, with emphasis on use in a commercial setting. In order to test the method a Norwegian company working with GIS on a daily basis which is considering adoption of FOS GIS will be used as a case study. There are several kinds of FOS GIS products available in different categories. In order to limit the scope of this study the evaluation and selection will be limited to the subset of GIS known as *desktop GIS*.

When considering adoption of FOSS there are some special considerations to make compared to adopting Commercial off-the-shelf (COTS) components (Cruz et al., 2006). Some requirements may also be domain-specific. This calls for a tailored method to evaluate GIS software. Several frameworks and articles on FOSS adoption will be analyzed in order to come up with a method for the evaluation of FOS GIS. The actual process of evaluating will be carried out in order to serve as an example on how to apply the method and to give an insight on the state-of-art when it comes to FOS desktop GIS.

The rest of this paper is organized as follows: In section 2 a brief history and definition of the term GIS are given, and the the term *desktop GIS* is investigated, pointing out what separates it from Web/Internet GIS. Section 3 gives an account

¹Free and Open Source Software for Geospatial Conference

of the *FOSS phenomenon*, presenting the history and describing common FOSS *licenses* as well as reasons for using Free and Open Source Software. Section 4 takes a look at *Free and Open Source GIS* (FOS GIS), giving an overview of organizations supporting development and exploring *who* develops and contributes to such projects. Relevant research on FOS GIS is presented to provide an insight in the ongoing research and developments in the FOS GIS field.

Section 5 examines several *frameworks for evaluating FOSS projects* for use in a commercial setting and describes a *tailored method* for selecting the right desktop GIS for a given case. Section 6 analyzes a commercial firm considering the use of a FOS desktop GIS by means of an interview. These findings are used in section 7 where the *method is applied* to select the best fit among ten candidate projects. The *results* of the evaluation are presented and a *final selection* is made in section 8. Section 9 presents thoughts on further work that can be carried out in order to *refine the evaluation process*. Finally this study is wrapped up in section 10, where the work is *discussed* and some *concluding remarks* are made.

2 What is a Desktop GIS?

The term “desktop GIS” is not well-defined, but it is still used to describe a subset of GIS. As this project deals with desktop GISes a definition is essential. The reader should have in mind that several definitions exist, and that no definition is the “right” one. Despite this there seems to be a consensus on what a desktop GIS is among users and developers. In order to establish this understanding a brief history and description of Geographic Information Systems (GIS) in general will be given. On this basis the terms “Web/Internet GIS” and “desktop GIS” will be examined, as they are both subsets of the GIS term. An overview of Web/Internet GIS is given as the borders are sometimes unclear, and the terms in some cases intertwine. The reader should also be aware that definitions may change over time.

The goal is to provide the reader with an understanding of the history, background and features of GIS in general and desktop GIS in particular. This will make it easier to understand the decisions and conclusions that are made later in this study. The reader should be aware that the overview given here just scratches the surface of the history of a discipline that has evolved over the last 40–50 years. The intention is not to give a comprehensive description of the topic of GIS, but to establish an understanding that will facilitate the evaluation of a specific type of GIS.

2.1 GIS - Geographic Information Systems

As stated a desktop GIS is a subset of the broader term GIS, and as such an understanding of GIS in general is essential to understand what makes a desktop GIS special and how it differs from other kinds of GISes. A brief account of the GIS history will be given, different definitions will be presented and on the basis of this some common characteristics of a modern GIS will be presented.

Many have tried to summarize the history of GIS in order to facilitate an understanding of what it is and how it can be used. Heywood et al. (2002, p.175–189) gives a rather thorough account of the GIS history, and this will form the basis for this summary. The development of what is known today as GIS started in the 1960’s with the CGIS² project and work by the US Bureau of Census. The trend was picked up at Harvard Graphics Laboratory and led to the launch of the first commercial GIS software in the seventies. The seventies also spawned ESRI, the company that is one of the leading GIS vendors today, as well as the first conferences and research work on GIS. The advances in computer technology in this decade assisted the developments in GIS, and several GIS products entered the market. ESRI launched their first GIS (ARC/INFO) in the early eighties.

²Canadian Geographic Information System

By the time of 1980 computers were used in all aspects of cartographic production, and demand for graphics, data-analysis, real-time querying of databases and topological overlays grew. Several disciplines were combined (computer cartography, GIS and spatial analysis) and formed the Geographic Information Systems of today. Another important event in the eighties was the launch of the microcomputers. This led to a new generation of GIS, the microcomputer GIS. In 1991 this trend continued when ESRI launched ArcView, a “desktop mapping and GIS tool”. The nineties was also the decade that gave birth to “internet mapping”, the first steps towards Web/Internet GIS.

2.1.1 Definition

A very basic definition of a GIS would be “an information system for geographic data”. This definition is however too vague to be useful, and a more specific definition is needed. While no definitive definition exists according to Heywood et al. (2002, p.11), several attempts have been made to define the term. Cox and Gifford (1997) gives an account of several such attempts, such as the definition presented by Aronoff (1989):

“A GIS is designed for the collection, storage, and analysis of objects and phenomena where geographic location is an important characteristic or critical to the analysis.”

and the definition by Cox and Gifford, (1994):

“[A GIS is] a computer tool for managing geographic feature location data and data related to those features.”

Barndt (1998) presents yet another definition:

“GIS software is a tool for processing information. Although GIS is often mistaken for map making, the *I* for *Information* is the most important part of the acronym.”

What these definitions seem to have in common is the perception that a GIS is a computer system used to handle geographic information, derive new information from it, manipulate it and present the results in some way. Some claim that a GIS does not have to reside in a computer environment (Cox and Gifford, 1997), but there is a common understanding that a GIS is a computer-based system. This assumption is made in this study.

As well as defining what is *meant* by GIS there have been several attempts to define the *features* of a GIS. Cox and Gifford (1997) summarizes the main GIS features as:

- Data input
- Data management and manipulation
- Data analysis
- Data output

This list of GIS features matches the features described by Heywood et al. (2002), and can be treated as a rather generic set of GIS base-functions.

From these definitions and features it can be derived that a GIS (in the broadest sense) is concerned with *what* one can do with spatial data, not so much with *how* it is done. The main GIS features described can be carried out by hand, on a desktop computer, or via a web-interface. This is made clear by the fact that while the computer technology has changed dramatically throughout the history of GIS, covering nearly 60 years, the general principles have remained relatively unchanged. The first generation GISes used mainframe computers with Command Line Interfaces (CLI) and devices such as map-plotters and scanners. The birth of the microcomputer led to development of on-screen map displays and cartographic animation supported by Graphical User Interfaces (GUIs). The birth of the Internet and the World Wide Web gave birth to web-based applications and standards for on-line transfer of spatial data.

Despite all of this, a GIS is still about input, management, manipulation, analysis and output of geographical data. The technological changes have altered what is meant by terms such as output (a print-out, or a zoomable on-screen map), but the basic principles remain the same. With this in mind the more recent concepts of Web/Internet GIS and Desktop GIS should be examined.

2.2 Web/Internet GIS

Internet mapping was launched in the 1990's (Heywood et al., 2002). The ESRI User Conference in 1997 dealt with the subject "Internet GIS" (Peng, 1997) and recognized that "The Internet is shaping the ways of traditional GIS function [*sic*]" . The Open Geospatial Consortium (OGC) released the first version of the Web Map Service (WMS) specification in 2000 (OGC, 2000b), and the Web Feature Service (WFS) specification in 2002 (OGC, 2002a). Together with the Geography Markup Language (GML) specification (OGC, 2000a), these specifications are an attempt to enable interoperability between different Internet GISes (Peng and Zhang, 2004).

In February 2005 Google launched Google Maps, an on-line mapping service with an easy user interface and the ability to produce "mashups" (Miller, 2006). This gave the broader audience an insight in GIS, and sparked an interest in

Web GIS functionality. This rise in public interest combined with the work on standards for Web GIS makes it clear that the Internet have brought changes to how the public relates to GIS, and how GISes are used and developed. Most current Internet GISes provide at least the basic GIS functionality discussed earlier, they are available almost anytime, anywhere and they have a reduced learning time (Peng and Zhang, 2004). The structure of a typical modern Web/Internet GIS is illustrated in figure 2.1(a).

While it may seem that a Web/Internet GIS can fulfill all the GIS functions, there is still a need for a more “traditional” GIS, as not all GIS tasks benefit from this new technology. The concept of desktop GIS has emerged to describe these desktop GIS applications in a world where Internet-based systems tend to be the standard.

2.3 Desktop GIS

A Desktop GIS is a description of a GIS that is installed on a desktop computer, and can be thought of as a *retronym*³ coined for what was traditionally considered a GIS. The term is not as well-established as GIS, but it is used in several research articles (such as Gray, 2008; Egenhofer and Kuhn, 1998; Souleyrette and Anderson, 1998; Strasser, 1995).

The major GIS company ESRI divide their GIS products in three categories; *Desktop*, *Server* and *Mobile* GIS and their *GIS Dictionary* (ESRI, 2008) defines *desktop GIS* as:

“Mapping software that is installed onto and runs on a personal computer and allows users to display, query, update, and analyze data about geographic locations and the information linked to those locations.”

This definition is adopted by Steiniger and Bocher (2008), adding: “That is, the software is not executed on a server and remotely accessed from or operated by a different computer.” The term desktop GIS is also used by the FOS GIS project *uDig*⁴ (an acronym for user-friendly Desktop Internet GIS). The software is described as: “Desktop located, running as a thick client”. A typical desktop GIS setup is depicted in figure 2.1(a).

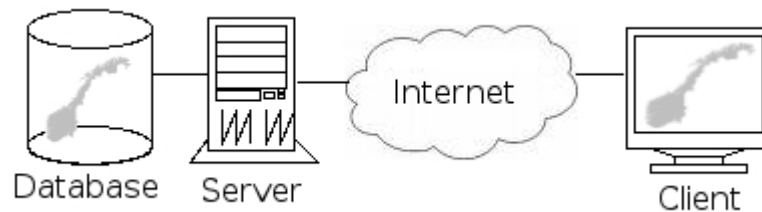
From these different definitions it is evident that one important aspect of a desktop GIS is, surprisingly enough, the fact that it runs on the desktop. This

³A retronym is a type of neologism coined for an old object or concept whose original name has come to be used for something else, is no longer unique, or is otherwise inappropriate or misleading (Wikipedia, 2008).

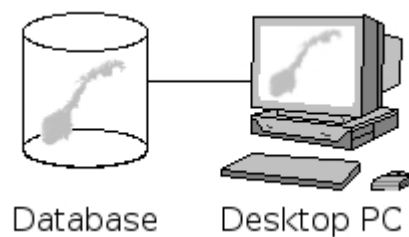
⁴<http://udig.refractions.net>

separates it from Web GIS, as seen in figure 2.1, and is more specific than the definition of a GIS. Still the term is rather vague, and it is not certain that ESRI and Refrations agree on whether a specific product is in fact a desktop GIS.

Compared to a traditional GIS, a desktop GIS can be seen as more concerned with what the user sees on his desktop, and less about data input, storage and management. A desktop GIS may fetch data from spatial databases or over the Internet with technologies such as WMS and WFS. Analysis, output and manipulation are tasks typically associated with the term desktop GIS. In this study a desktop GIS will be defined by the ESRI definition, adding that it may support fetching of data over the Internet.



(a) Web/Internet GIS



(b) Desktop GIS

Figure 2.1: A comparison of typical setups for Web/Internet and desktop GIS. The Desktop GIS is located closer to the data source, but can also fetch data over the Internet.

3 What is Free and Open Source Software?

Free and Open Source Software (FOSS) is one of several terms used to describe a particular category of software. What characterizes this kind of software is that it is distributed under a *license* that grants the users several *freedoms*. The concept is in opposition to *proprietary* software, where the developers have the exclusive rights to the source code, and users only have access to a *binary* version of the software. Proprietary software usually comes at a price, and with a license that prohibit the users to redistribute and change the software. The FOSS licenses, on the other hand, ensures that the users have the freedom to:

- Use the software (for any purpose)
- Modify the software (and distribute their modifications)
- Redistribute the software

without having to pay the software author(s) a fee or royalty to do so (Feller et al., 2005).

This concept represents a radical change compared to the way the software industry usually works. In the later years this way of distributing software has gained popularity, and in order to understand how commercial companies can benefit from this paradigm a closer examination of the phenomena is needed. This section will briefly explain the history of FOSS, describe some of the most popular FOSS licenses and look at reasons to use Free/Open Source Software from a commercial point of view.

3.1 Free Software and Open Source Software

The term Free and Open Source Software seems a bit redundant, and in order to understand the term one have to look at the history of *Free Software* and the later term *Open Source Software*. Both terms, as well as the term FOSS, are in use today, and from a pragmatial point of view they have the same meaning (i.e. software that fulfills the above mentioned requirements). This historical overview is not meant to be complete, as the history of FOSS is covered in detail by others (for example Lerner and Tirole, 2002), but is an attempt to establish an understanding of the *reasoning* behind this software paradigm and an understanding of the different terms.

In the early 1980s Richard Stallman (then a programmer at the AI Laboratory of MIT) founded the Free Software Foundation (FSF) and coined the term *Free Software*, as defined in the “The Free Software Definition”⁵. The main reasoning

⁵<http://www.gnu.org/philosophy/free-sw.html>

behind this incentive was the belief that source code should be free and skepticism towards the commercialization of the software industry. This attitude grew from the academic and “hacker” cultures of MIT and Berkley in the advent of computer science. While the English term “free” is ambiguous, the FSF makes it clear that the term *Free Software* refers to “free speech”, rather than “free beer”.

Later on, in 1998, Netscape released the source code for their Netscape Navigator Internet browser as FOSS. At the same time Eric Raymond and others coined the term *Open Source* (Raymond, 1998) and founded the Open Source Initiative (OSI), with their own “Open Source Definition”⁶. The focus of the OSI is commercial adoption of FOSS and the software development process used, rather than the ideology of the FSF.

The main difference is thus that the FSF is more ideological (Szczepanska et al., 2005), while the OSI is younger with a more pragmatic approach (Lerner and Tirole, 2002). As illustrated in figure 3.1 the terms “Open Source Software” and “Free Software” overlap, and all the typical FOSS licenses are approved by both organizations (Steiniger and Bocher, 2008). The term FOSS is thus a bit redundant of historical reasons. The main difference between the FSF and OSI is the *motivation*, rather than the *result*. For the pragmatic user or developer the difference is more in the *license* than the *organization*, and thus the term FOSS will be used to describe all such software in this study, and specific licenses will be described where relevant.

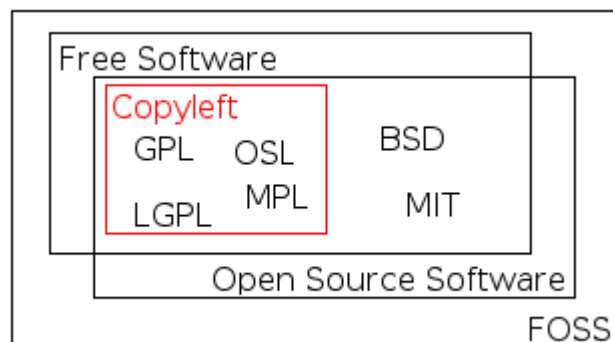


Figure 3.1: *Free Software and Open Source Software illustrated (based on the figure by Chao-Kuei, <http://www.fsf.org/licensing/essays/categories.html>).*

⁶<http://opensource.org/docs/definition.php>

3.2 Licenses

As stated, the main difference between FOSS products is the *license* they are released under. A FOSS license is a license designed to preserve the freedoms described in the beginning of this section, and they are typically *approved* by the FSF and OSI. These licenses can be grouped in three broad categories, as listed in table 3.1. In addition to these categories there are some licenses that lacks approval, but still claim to be FOSS licenses. Fitzgerald (2006) refers to these as “Shared-Source” licenses.

License category	Examples
Reciprocal	GPL, LGPL, OSL
Academic Style	Apache License, BSD, MIT, ...
Corporate Style	MPL, Qt Public License, ...

Table 3.1: *The three FOSS license categories, and examples of each category.*

The *reciprocal* licenses are also known as *viral* licenses. Their main goal is to support the notion that software should be free (as defined by the FSF). The main license in this category is the GNU Public License (GPL), which employs the principle of *copyleft*. This means that all programs containing GPL code is required to be released under GPL themselves. The GNU/Linux Operating System is one of many products licensed under the GPL (Stallman, 2007).

Academic style licenses impose few restrictions, the main requirement is that one acknowledge the work of previous contributors. The Berkeley Software Distribution License (BSD) is the most used license in this category. Apples OS X uses the BSD licensed BSD system at its core (Sánchez, 1999).

The third group of licenses is the *Corporate Style* licenses. These licenses are targeted at corporate interests rather than the FOSS developer community, and allow mixing of FOSS and proprietary code and focuses on letting the corporate sponsors have control over derivative works. The web browser *Mozilla Firefox* is the best known product licensed under the MPL.

One concern with FOSS licenses is the fact that since anyone can modify the programs and distribute their modifications a project may end up with a lot of *forks*. The F.F.I.E. Council (2004) defines forking as follows: “A fork occurs when the development community splits over the path of development of a given application. In the worst-case scenario, development of forked FOSS may be halted, or the technical direction may become so altered that it no longer meets the institution’s needs.”

3.3 Why use FOSS?

Why should a company use FOSS instead of proprietary software? How does a commercial company benefit from using FOSS products? These are essential questions when considering the use of FOSS in a commercial firm. The reader should be aware that the reasons for *using* FOSS products generally differ from reasons to *develop* FOSS products, which is a study in itself (see for example Ye and Kishida, 2003; Ruffin and Ebert, 2004).

FOSS advocates (such as Wheeler, 2004) tend to point to well-known and successful FOSS-products being used worldwide when asked why one should adopt FOSS products. However, the success of the operating system GNU/Linux or the web-server Apache can not reveal much about the advantages of FOSS GIS. Of course it is likely that they use the same (or a compatible) license, but that is about it. Different FOSS projects are developed in different ways, by different people with different expertise and varying degrees of support and funding. On top of this the domain-knowledge varies from project to project. Based on this one cannot say that “FOSS product X is really good, therefore you should start using product Y, which is also FOSS” (Golden, 2005).

However, an understanding of the generic reasons companies have for using FOSS products can probably give some insight to the growing popularity of FOSS in the commercial sector. A Norwegian study (Hauge, 2008) takes a look at the use and integration of FOSS in the Norwegian software industry. Thus the results mainly consider usage of FOSS components in software development. Despite this the results may shed light on why companies in general use FOSS. The study revealed that about half of the 700 examined Norwegian software companies used FOSS products in their solutions, and the three main reasons for using FOSS components where:

1. Easy access to (in many cases unique) functionality of high quality.
2. No license costs reduces total development cost
3. Re-use of already tested components enhances the quality of the final product.

At least reason 1 and 2 can be applied to using a FOSS product such as a desktop GIS with no software development involved. These findings also match the reasons listed by OpenBRR (2005) for considering FOSS: Cost, source code access, open architecture and quality.

From this (limited amount of) data it is evident that there are a number of reasons to choose FOSS software, but what is clear is that the ideological aspect (as endorsed by the FSF) is not important for companies. In general companies tend to focus on price and quality. This is exemplified by the Norwegian Mapping

Agency. They switched to GNU/Linux as the operating system for their on-line map servers. They state that this has increased functionality and stability, in addition to savings of 2 million NOK, which is considered a nice side effect (Dagens It, 2008).

The bottom line should be that there are as many reasons for choosing FOSS products as there are companies, no set of reasons matches all companies and situations. Business area and available resources play a major part in the decision. As a consequence one must carefully examine the FOSS products in question before using them in a commercial setting. A tailored method for evaluation of FOSS products in a GIS setting is described in section 5.

4 FOS GIS Overview

Free and Open Source GIS (FOS GIS) (Norwegian: FriGeoProgramvare) is, as the name implies, GIS software released under a FOSS compatible license. A rather large community developing, using and promoting FOS GIS has emerged in the later years, with commercial companies, governmental institutions and private initiatives contributing money, skills and code to a variety of different projects. To get a quick overview of the diversity of FOS GIS projects the Open Source GIS webpage⁷ can be examined. The page currently⁸ lists 247 GIS related projects. While this number includes projects not strictly GIS (such as image manipulation software) it gives a rough estimate, and an idea of what kinds of GIS related software the FOSS community is developing.

This section will try to give an overview of the current status of FOS GIS. Organizations working to promote and organize FOS GIS projects and other relevant initiatives supporting the development, and use, of these projects will be examined. A description of *who* provides such software as well as the reasons *why* they do so is also given. Existing research on FOS GIS, both evaluations of specific projects and more general analysis, will be examined and the main conclusions presented. This will help give the reader an insight in FOS GIS and how it differs from the proprietary products that until recently have been the standard way of dealing with GIS. Although both open standards and free geospatial data are important issues, the main focus is the software, and these subjects will not be dealt with unless there is a strong correlation with the software.

4.1 Organizations

Since its founding in 2006 the international foundation “The Open Source Geospatial Foundation” (OSGeo) has been acting as a central node in the FOS GIS community. They serve as a host for several projects, (see table 4.1), in addition to hosting the annual FOSS4G conference and publishing the OSGeo journal. As they state on their webpage⁹:

“The foundation provides financial, organizational and legal support to the broader open source geospatial community. It also serves as an independent legal entity to which community members can contribute code, funding and other resources, secure in the knowledge that their contributions will be maintained for public benefit.”

⁷<http://opensourcegis.org>

⁸As of 21.10.2008.

⁹<http://www.osgeo.org/content/foundation/about.html>

Type	Examples
Web Mapping	Mapbender, MapBuilder, MapGuide Open Source, OpenLayers
Desktop Applications	GRASS GIS, Quantum GIS
Geospatial Libraries	FDO, GDAL/OGR, GeoTools

Table 4.1: *Examples of OSGeo hosted projects, taken from <http://www.osgeo.org/>*

On the more informal side several of the “high profile” FOS GIS projects cooperate and share ideas on many levels. For instance the Canadian company Refrations Research develops both the spatial database extension PostGIS and the desktop GIS uDig, in addition to contributing to the web map server MapServer. The developers of uDig also cooperate with the team who develops GeoTools, the geospatial library hosted by OSGeo. Geospatial libraries such as GeoTools, GDAL/OGR and JTS are used by several other FOS GIS projects. This creates a shared network of FOS GIS developers that reaches beyond the boundaries of the different projects, and creates a “multiproject software ecosystem”, as described by Scacchi et al. (2006). The result is that the projects benefit from shared resources and communication, increasing the efficiency of development for all the involved projects.

In addition there are local or national incentives to encourage the use of FOSS in general, and in some cases FOS GIS in particular. In some countries, such as Canada and Spain, projects not only to use, but also develop FOS GIS products have been initiated by governmental organizations. Examples are the original JUMP project and the gvSIG project¹⁰. The reasons for this can be economical or political or a combination. In Norway a “national resource center for FOSS” (Friprogcenteret)¹¹ was opened in August 2007. One of its stated goals is to: “Stimulate governmental and private businesses to establish a sharing culture and use free and open source software”. This center has, among other things, initiated “Delingsbazaren”¹², a webpage that aims to enable governmental organizations to share FOSS projects developed with governmental funds.

Currently only one downloadable GIS-related project (sosi2kml) is listed in Delingsbazaren, although the National Mapping Agency is listed with intentions of sharing. They do not provide any downloads but they promote the webpage frigeoprogramvare.org. This is an on-line forum and a wiki for discussing the use of FOS GIS in Norway and is open to both private and governmental organizations. The webpage was established after a Norwegian workshop on geospatial FOSS

¹⁰See section 7 for a description.

¹¹<http://www.friprog.no>

¹²<http://www.delingsbazaren.no>

in March 2008. The Mapping Authority established the webpage with support from “Norge Digitalt”, a collaboration between enterprises working with spatially referenced data in Norway. Although this seems to indicate that there is an interest in FOS GIS in Norway, there is little activity on the webpage. The discussion board has a total of 66 entries and 45 registered members. The wiki is even less used, with less than ten articles in total¹³. One reason for this may be that the website only deals with user-to-user support and does not host or develop any projects on its own.

4.2 Sources and Sponsors

Both commercial firms and governmental institutions contribute to the development of FOS GIS. Steiniger and Bocher (2008) points out that one possible grouping of FOS GIS projects is by who funds and maintains them. They identify three such groups; commercial firms, enthusiasts or community projects, and governmental or educational institutions. Table 4.2 shows examples of some FOS GIS projects and what their sources are (i.e. who initiated the project).

When comparing this table with the projects hosted by OSGeo (table 4.1) it is evident that several projects started by governmental or educational institutions have been adopted by OSGeo. The GRASS history may serve as an example; The GRASS project was initiated in 1982 as a U.S. Army Corps of Engineers CERL¹⁴ project, and after some years with decreasing interest and funds it was transferred to a non-profit organization in 1997. In 1999 it was released under the GPL license, and is now an OSGeo project (Mitasova and Neteler, 2004). This project was at first developed for governmental purposes with governmental funds, and only later did it adopt a FOSS license, “giving back to the tax-payers what they have paid for in the first place” as Steiniger and Bocher (2008) puts it.

Another example is the JUMP (Java Unified Mapping Platform) project that was initiated as a collaboration between governmental and commercial interests in Canada. Lack of funding brought the development to a halt in 2004. After a while a group of volunteers founded an organization to continue the development under the name OpenJUMP. OpenJUMP decided in December 2008 to apply for OSGeo Project status¹⁵.

These examples show that a living community and funds are vital for a FOSS project to prosper. This can be achieved in several ways; one is funding by commercial sponsors, another is governmental funds. In some cases a community-driven project is able to withstand, but the fact that several community-initiated projects

¹³All numbers gathered on 28.10.2008.

¹⁴Construction Engineering Research Laboratory

¹⁵http://groups.google.com/group/openjump-users/browse_thread/thread/280d8ea0f925b59c

are (or are applying to be) OSGeo projects shows that in any case a strong community and channels for promotion are needed. Third parties considering the use of FOS GIS should be aware that projects funded by governmental institutions may have a set of guidelines and goals that are controlled by politics and the needs of the sponsoring governmental institution. Because of this it *may* be more difficult to influence the development.

Source	Examples
Government/Educational	GRASS (U.S. Army), Mapserver (University of Minnesota, NASA)
Commercial	PostGIS & uDig (Refractions), KOSMO (SAIG S.L.)
Community/Private	GDAL, Quantum GIS

Table 4.2: *Different FOS GIS products ordered by source. (See appendix F for links to history/explanations.)*

4.3 FOS GIS Research

When it comes to existing research on FOS GIS these typically falls into two categories: Evaluation of GRASS and its history, explaining the shift to a FOSS strategy and how both the project and the GIS community can benefit from it. The other topic is the use of different FOS GIS products as part of setting up a GIS “stack”. Typically these studies evaluates web map servers such as GeoServer and MapServer and spatial databases like PostGIS.

Mitasova and Neteler (2004) gives an overview of the GRASS history, describes its development method and the evolution of its functionality. They also present some thoughts about the future of GRASS, emphasizing the opportunities that lies in the improved structure, modularity and higher level of abstraction that came as a result of the FOSS adoption. The paper concludes that: “Open Source and Free Software infrastructure offers an excellent opportunity for efficient development of a robust and reliable GIS code with freedom for everybody to run, study, redistribute and improve it for the benefit of the entire community.”

Löwe (2008) also takes on GRASS, focusing on the positive aspects of it being *legacy* software, among these is its ability to serve as a backend for other GIS products such as QGIS and uDig. On the other hand he notes that “A tool to describe processing chains would be desirable to document and manage the community-inherent know-how about how to orchestrate the GRASS modules to cope with complex tasks.”

Other GRASS papers include Neteler and Raghavan (2006), which focus on GRASS 6, and notes a special benefit of FOS GIS: availability of the underlying

(spatial) algorithms. Yet another GRASS study is the masters thesis by Buchanan (2005), in which he compares GRASS 6.0 and ESRI's ArcGIS 9.0 and concludes that GRASS is an effective alternative.

It may seem that the study of FOS desktop GIS suffers from the same problem as general FOSS research does according to Østerlie and Jaccheri (2007), which claims that most Open Source Software development studies tend to focus on well-known projects such as GNU/Linux, Mozilla and Apache and that this leads to a research bias. In the same manner the large amount of GRASS studies can be considered a bias. This may not be the case, another explanation can be that there are few FOS desktop GISes as well-known as GRASS.

At least one study seems to break this GRASS “bias”. The article “Overview on current FOS Desktop GIS Developments” by Steiniger and Bocher (2008) is already mentioned. This paper presents ten desktop GIS projects, and provides a basic description of each project (with more thorough information and tables provided on a webpage). They also discuss several pros and cons of FOS GIS from a general and an university perspective, emphasizing, as Neteler and Raghavan (2006) does, that the availability of the underlying algorithms is an important aspect. They also emphasize that the adoption of FOSS must be carefully considered and that even though the software is free a switch is not without cost.

Ramsey (2007) presents a survey on Open Source GIS, presenting a total of 33 FOSS GIS projects, grouped by the programming language they are implemented in (C, Java and .Net) as well as making a distinction between *libraries* and *applications*. He concludes that the FOS GIS space includes products to fill all the slots in an GIS stack, as shown in figure 4.1

Câmara and Onsrud (2003) conducted a survey of 70 FOS GIS projects and analyzed who developed them and whether they could be considered *innovative*. They show that corporations are the main developers of successful FOS GIS software, and that networked teams of individuals develop only a small percentage of the FOS GIS projects. The direct involvement of universities is found to be limited. This may be explained by the fact that successful FOSS projects require long-term involvement on maintenance and upgrades. Universities tend to focus more on developing new ideas. An interesting observation is that a rather small proportion of the projects are considered innovative. This may be explained by the fact that FOSS is mainly used as a way to reduce costs and break the commercial monopolies in the GIS sector. The authors also indicate that corporations will develop software based on their strategical needs. As a result of this, governments with particular requirements need to establish public-funded projects to ensure that the software will meet their needs.

An example of a study of FOS GIS with the aim of establishing a FOS GIS stack is the report by Ahearn et al. (2006) for the United Nations Office for Coordination

of Humanitarian Affairs. The study included the setup of a proof of concept FOS GIS stack, with a data, web, client, and web services tier. Several FOS GIS products are evaluated and described, such as PostGIS, GeoServer, JUMP and uDig as well as Google Earth, which is free but not FOSS. The conclusion is that both the *data* and *web services* tiers have mature FOSS candidates, while the *client* tier (both thin and thick clients) “needs significant development before it can realistically be used to fulfill [the needs of the project]”, noticing that the primary weak points are data editing and cartographic output.

Another study that reinforces the claim that the database part of FOS GIS can be considered mature is presented by Wikstrøm and Tveite (2005). Their study compares PostGIS and MapServer to ESRI's ArcSDE and ArcIMS, and concludes that the FOSS solution can match the proprietary ESRI stack, and that “if the development of this type of Open Source software continues in the same manner, there is no reason not to choose this as an alternative to commercial solutions in the future”.

Moreno-Sanchez et al. (2007) explores the use of several FOS GIS components for setting up a cross-border health study with spatial elements. The study describes a pilot project that involved setting up a WBMMGIS¹⁶ using FOSS and open standards. The FOS GIS components used were PostGIS and MapServer. The study focuses more on political and sociocultural aspects than on technical performance, and concludes that if FOS GIS products are to be used in a mission-critical context research has to be carried out, both with regard to performance and sociocultural aspects.

The examined papers seem to agree that although some FOS GIS projects are matching, or even outperforming their proprietary counterparts, there is still a need for further research on these projects. There seems to be a notion that FOS GIS yet has to “prove itself worthy”, and while some researchers seem to be convinced that FOSS is the future they have little statistical backing for this claim. And again, as Østerlie and Jaccheri (2007) points out: all FOSS projects are different. There is a risk in drawing conclusions on FOS GIS in general based on a small set of studies of a limited number of projects.

One trend worth noting is however the findings by Ahearn et al. (2006), that the client tier was the weakest part. The cause for this may be the fact that the FOSS community traditionally have focused on developing server-side products, such as databases and operating systems (think of GNU/Linux and Apache), but desktop applications that require more usability considerations have been neglected until recently. This claim is backed by Nichols and Twidale (2002) who analyzes usability in FOSS. However it should be noted that the study by Mitasova and Neteler (2004) is two years old, and that things change rapidly in the software

¹⁶Web-based multimedia spatial-information system

industry. The more recent study by Steiniger and Bocher, concentrating on the client part, does not mention any specific usability problems.

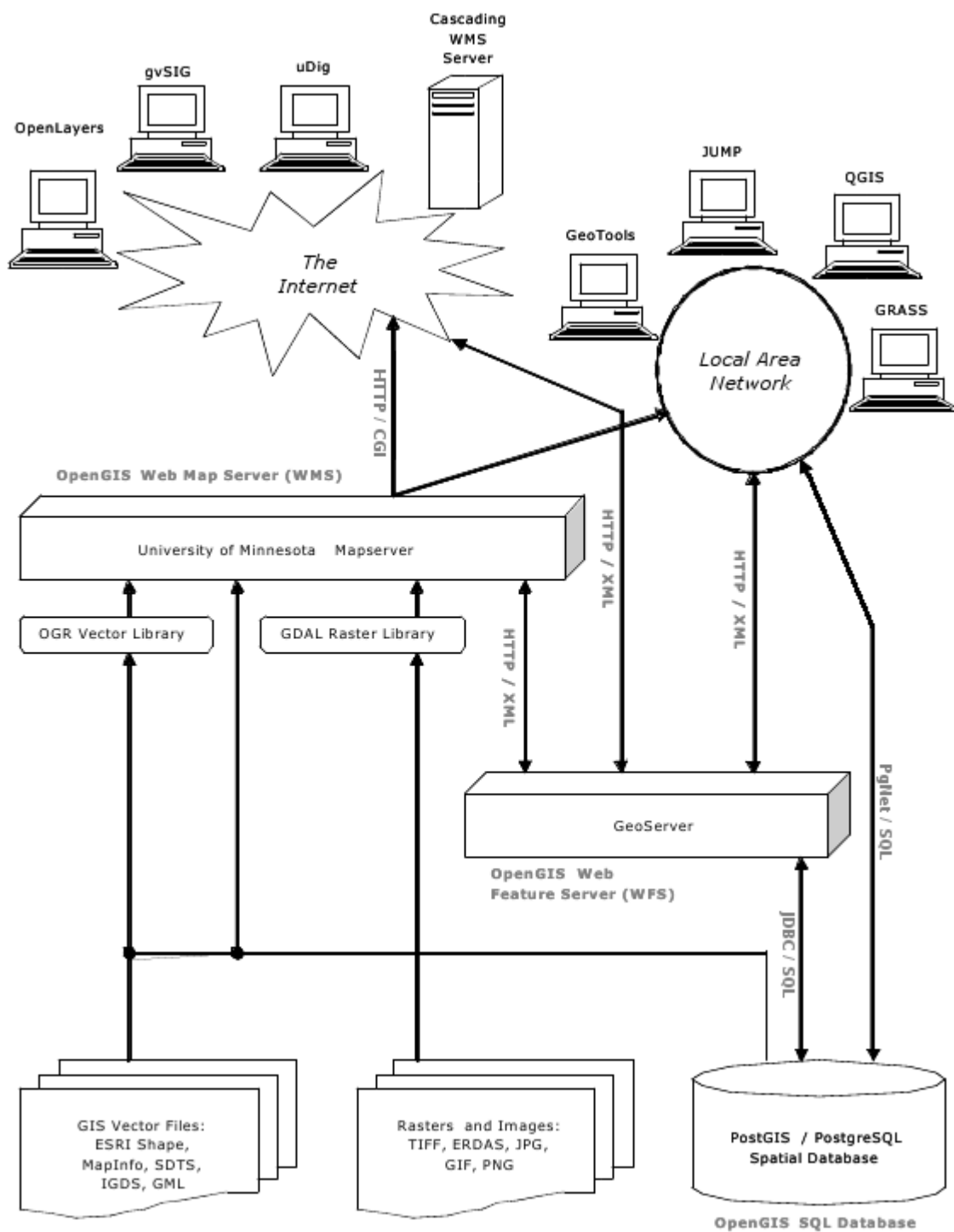


Figure 4.1: The OpenGIS stack as presented by Ramsey (2007).

5 A Method for Evaluating FOS GIS

Section 3 concluded that one must carefully examine the FOSS products in question before using them in a commercial setting. In order to ensure that all relevant aspects are considered some sort of formal method or framework can be useful. Cruz et al. (2006) notes that since FOSS differs from traditional COTS software the methods used to evaluate COTS may not be applicable to FOSS.

This section will analyze the need for a formal method for selecting FOSS by reviewing relevant studies on FOSS adoption. Three frameworks designed for evaluation of FOSS products will be examined. Based on these a tailored method for selecting the right FOS desktop GIS will be formulated. This method will be applied in section 7, and will serve as an example on how to apply the tailored method.

5.1 Why use a Framework?

The Norwegian study by Hauge (2008) found that evaluation of FOSS is usually conducted in a rather informal way. The evaluation is normally carried out by a single developer, using no formal method or framework. This developer tends to choose FOSS products he knows from previous experience or in other ways trust. When evaluating products they have no previous experience with companies usually carry out activities such as inspection of developer documents and reference implementations.

The study points out that although most companies use this informal approach, process frameworks and more formal reference implementations are used by some companies. None of the participants in the study reported using a FOSS evaluation framework. There can be many reasons for this, one possible explanation is that these frameworks are perceived as rigid and not flexible enough. Another explanation is that since this study is concerned with FOSS as components in software development, the individual components are small and well known to the developers in the beginning. The components used may also be considered a de facto standard, such as the zlib compression library¹⁷. This means that a formal process is seen as wasted effort. In this particular case the FOSS in question is a complete software system, and there are several candidates to choose from. This, combined with the fact that the company used as an example in this study has little experience with the candidate products, calls for a more formal evaluation method.

A formal evaluation method ensures that a broader range of alternatives are evaluated and it reduces the risk of prejudice from the evaluator. In addition,

¹⁷<http://www.zlib.net/>

using a formal approach forces the company to review their reasons for choosing FOSS in the first place, which can be a clarifying exercise. This is the motivation for this framework review and the construction of an evaluation method.

5.2 Existing frameworks and Literature on FOSS Evaluation

In this section three freely available FOSS evaluation frameworks will be reviewed; *Business Readiness Rating for Open Source* (OpenBRR, 2005), *Method for Qualification and Selection of Open Source software* (Atos Origin, 2006) and *The Open Source Maturity Model* (Golden, 2005). In addition an article by Cruz et al. (2006) on FOSS evaluation will be examined.

5.2.1 Business Readiness Rating for Open Source

The Business Readiness Rating (BRR) for Open Source Framework (OpenBRR, 2005) is a framework developed by Carnegie Mellon University and Intel. It promises to “...weigh the factors that have proven to be most important for successful deployment of open source software in specific settings”.

The framework relies on a method to standardize and group evaluation data in categories, and this assessment process is organized in four phases (see figure 5.1(a)). The execution of these phases leads to the calculation of a Business Readiness Rating (BRR) that reflects the most essential categories of the assessment.

The first phase, Quick Assessment, is concerned with finding candidate components, defining initial user requirements and removing the components that does not fulfill these requirements. The second phase, Target Usage Assessment, is about ranking a set of categories based on importance and give each component an importance factor and defining metrics within each category and ranking these as well. In the third phase, Data Collecting and Processing, data is gathered and values are assigned to the different metrics. The BRR is calculated in the last phase, Data translation. This BRR should provide a clear answer on what component to choose.

The framework supplies a descriptive list of 12 categories and a representative set of metrics (2–3 per category). The reader is encouraged to look up these in the paper describing the framework (OpenBRR, 2005).

5.2.2 Method for Qualification and Selection of Open Source software

The Method for Qualification and Selection of Open Source software (QSOS) (Atos Origin, 2006) is published by Athos Origin under an open license. It aims to

be a tool to “precisely examine the constraints and risks specific to open source software” and “differentiate the quite often numerous candidates...”.

The framework is made up of four interdependent steps that can be executed in iterative loops (as illustrated in figure 5.1(b)) to refine the process. The first phase, Definition, is about defining reference frames. This is not specific to any product, but to FOSS as a whole. Software families, types of licenses and communities are defined and used as a basis for further investigation.

The next phase is Evaluation. In this phase the software is evaluated, an *Identity Card* consisting of factual data such as name, type, licenses, documentation and comments is made for each candidate product. This data is not rated, but forms the basis for the rating on the *evaluation sheet*. For each software release examined an evaluation sheet is made, with criteria scored from 0–2, and applied on three axes: Functional coverage, Risks from the users perspective and Risks from the service providers perspective. It should be noted that the scoring is done independent of user context.

The third phase, Qualification, is about stating user needs. The user selects a desired requirement level on each of the axis of the functional grid. Filtering can also be carried out based on the identity cards, for example by selecting software compatible with a given operating system. The result of this phase is an ID card and a functional grid filter.

The last phase is selection. Two methods can be applied: Strict or Loose. Strict selection removes software not passing the ID card or functional grid filter in addition to software that fails to meet the users preference in the “User risk” category. The problem with strict selection is that it, due to its strict nature, may filter out all the candidates. The Loose selection method classifies the software and measures gaps, by assigning weights to the requirements (functional and User risk).

This framework presents a rather large set of criteria that can be used to form the evaluation sheet. These criteria covers “Risks from the users perspective” and “Risks from the service providers perspective”, criteria for functional coverage depends on the software family and is thus not listed. The QSOS webpage¹⁸ lists criteria for some software families, but not for the GIS software family.

5.2.3 The Open Source Maturity Model

The Open Source Maturity Model (OSMM) (Golden, 2005) is a framework by Navica, a consulting firm on open source. The aim of the framework is to answer the question “will an open source product, based on a company’s unique requirements, satisfy their needs?”

¹⁸<http://www.qsos.org>

As the name implies the model is used to determine the maturity of a FOSS product. The claim is that pragmatic organizations have two requirements for software: Mature products, that is, products that are of high quality, and that the product is fully functional. It is also desirable that the products are bundled with other product elements that make them easy and efficient to run. There are very few FOSS projects that fulfill these requirements, so companies have to identify those products that meet the maturity level required for their intended use.

The OSMM framework consists of three phases, providing a formal set of assessment criteria. In the first phase key product elements are identified and defined and the maturity of each element is assessed. Then a maturity score (between 0 and 10) is assigned to the product as a whole. In the second phase the elements are given weights (either default or custom). Using the maturity score and the weights an overall maturity score is assigned, using a scale from 0 to 100.

The paper describing the framework (Golden, 2005) presents a list of key product elements and default weights, as well as a table of recommended minimum OSMM scores for different types of users.

5.2.4 Evaluation Criteria for FOSS Products Based on Project Analysis

This paper by Cruz et al. (2006) presents a systematic approach for supporting the choice of FOSS products in an enterprise setting. The article provides a list of 11 different usage scenarios for FOSS, and describes requirements related to each usage scenario, grouped in six categories. Each requirement is presented and an explanation of why it is relevant is given. What investigable data that can be used to assess the requirement and how to gather this data is discussed. Finally a discussion on how to estimate whether a FOSS project fulfills these requirements is presented.

The authors conclude that "...the presented approach is not and cannot be understood as an automated decision system for the usage of F/OSS products in a company". This is in contrast to the three previous frameworks that present systematic approaches that will eventually produce a final set of "best fit" products. Although this strategy may seem tempting, it is important to be aware that human assessment is essential even if a set of scores is used. A set of guidelines as presented by Cruz et al. leaves more of the decision process to the user himself. This approach should not be underestimated and will be important in the tailored method.

This article is the only suggesting methods for gathering measurable data. This makes it a powerful tool for evaluating FOSS products, although the method can take longer time and leave more to the user to decide. The proposed strategies for gathering measurable data and the list of usage scenarios and related requirements

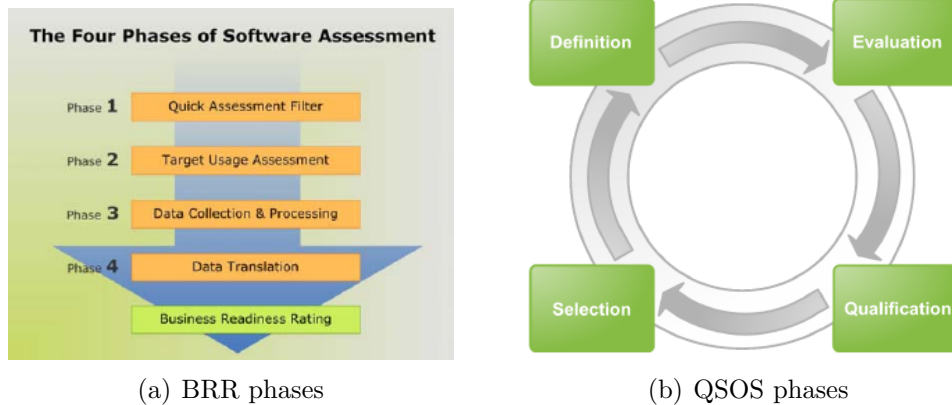


Figure 5.1: *The different phases of the BRR and QSOS frameworks illustrated, adopted from OpenBRR (2005) and Atos Origin (2006).*

provides a good starting point for the creation of a tailored evaluation method for FOS GIS.

5.3 A Tailored Method

When reviewing these frameworks and approaches it is evident that there are several ways to evaluate and find the right FOSS product for a given task in a given company. All the three frameworks use a step-wise approach (as can be seen in figure 5.1) that can be reduced to a generic approach consisting of four steps:

- Gather candidate projects and initially filter out ones that does not match
- Gather data from the selected candidate projects
- Define a set of evaluation criteria
- Evaluate the product according to evaluation criteria (and computing a score)

This approach will be used in the tailored method for selecting the “best” FOSS desktop GIS. Compared to the generic approach the tailored method will however start by defining evaluation criteria, as done in BRR. The usage scenarios presented by Cruz et al. (2006) will be used to identify requirements and evaluation criteria related to these. In addition an interview will be carried out with the company in order to get insights on functional requirements and to fine-tune the other (legal, economical etc.) requirements.

The next step in the tailored method is to gather candidate projects. To limit the scope of this study the ten FOS desktop GISes presented by Steiniger and Bocher (2008) will be used as candidate projects. Their list is rather exhaustive and cover most of the relevant candidates. In cases where such lists are unobtainable a survey will have to be carried out in order to identify candidate projects. Webpages such as the Open Source GIS webpage mentioned earlier could be used as a starting point for such a survey.

Based on the list of candidates the initial filtering can be carried out. The functional requirements serves as the filter, as any candidate that fails to meet these will be useless for the case in question. The concept of *Identity Cards* and *Evaluation Sheets* from the QSOS framework will then be used to provide an overview. The methods presented by Cruz et al. (2006) will be applied to gather information needed to evaluate how the candidate projects fulfill the criteria that make up the different requirements.

This will lead to a textual description (the ID card) and a set of ranked criteria (the Evaluation Sheet), describing how the candidate projects fulfill the specified requirements. Based on this a manual selection will be made, considering both the scores and the textual description. This assures that the selection is based on human assessment. The result is a rated and explained list of recommendations for a FOS GIS that meets the intended usage scenarios of the company.

This tailored method is illustrated in figure 5.2, and consists of the following four phases:

1. Determine usage scenarios and requirements
2. Gather candidate projects and perform an initial filtering
3. Create an identity card and evaluation sheet for the remaining candidates
4. Rank the projects and manually make a selection

The method described here will be the basic principle used in the assessment of candidate projects. It is important to note that although the method rely on a set of stated criteria that are ranked, these are only guidelines for the final selection. Human assessment is, as discussed, an important factor, and this is the reason that the method relies on a final, manual selection based on scores rather than a mathematical selection based purely on the obtained scores.

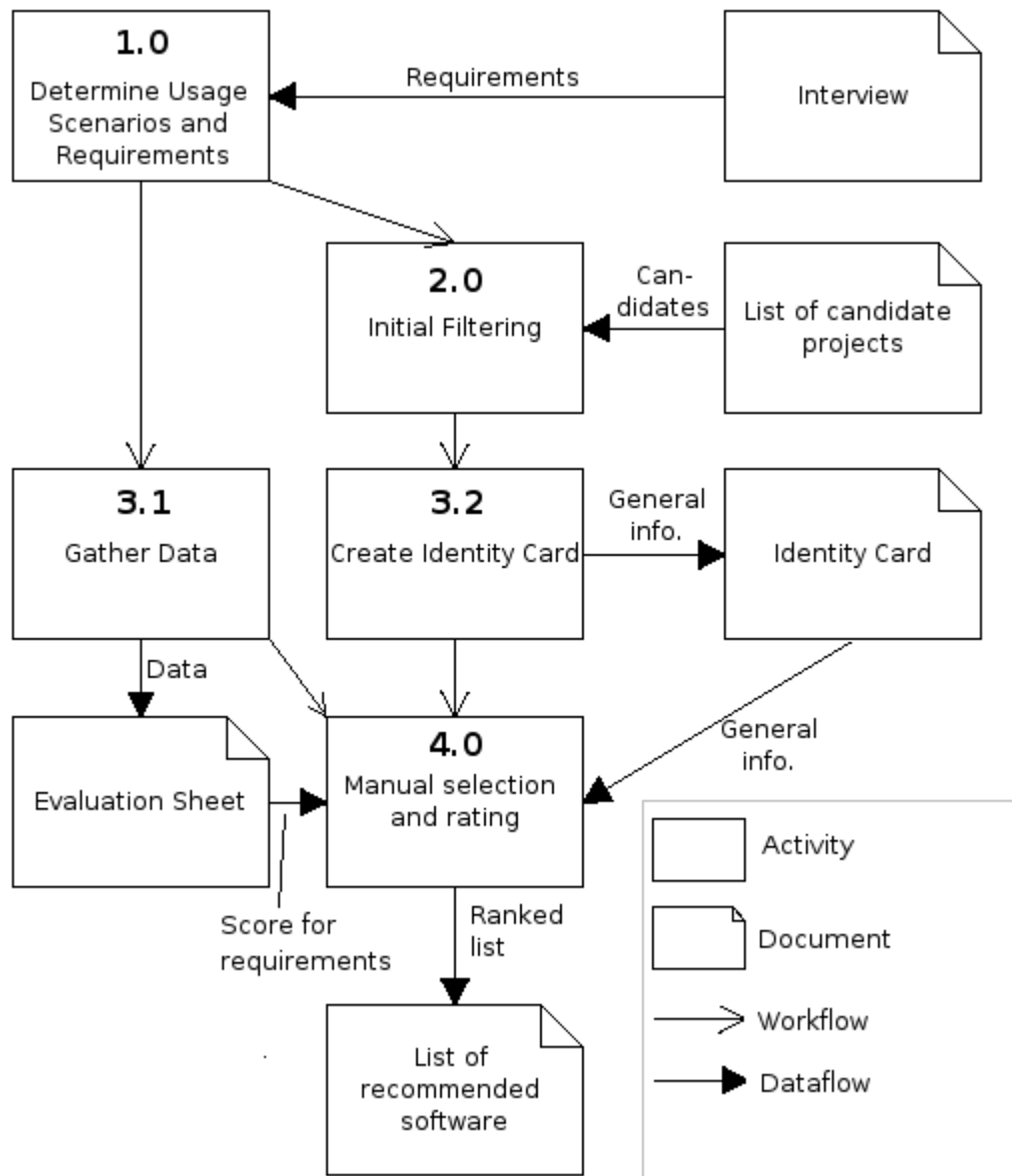


Figure 5.2: Evaluation method for FOS GIS illustrated, notice the difference between work- and dataflows as well as activities and documents.

6 Case Study

This section is intentionally left out by request from the company.

The content of this section was an analysis of the system used in the company, made through an interview-session (what is called a “structured focus group” by Marakas, 2006, p.91) with three employees at the company in question. For the question-set used see appendix A (presented in Norwegian).

The conclusions where the following:

To try and sum up what the company wanted with FOSS they where presented with 9 of the 11 usage scenarios presented by Cruz et al. (2006) (see section 5) and asked to specify to what degree they matched their thoughts on FOSS usage for the system examined. The results are shown in table 6.1. When it comes to research purposes it should be noted the company does no formal research, although several employees are interested in and are testing FOSS products on a day-to-day basis.

Usage scenario	Relevant?
Platform for a mission critical process	Not initially
As a long-time consideration	Yes
As a cost-reduction model	Yes
Exploration object (e.g. for technology)	This will be a consequence, not a reason
As a base line for further development and business model	Base line: probably, business model: not in the near future
Bridge a temporary bottleneck	No
Becoming independent of proprietary solutions and vendors	Yes
Gain transparency concerning safety and security	No
For research purposes	No

Table 6.1: *FOSS usage scenarios for the analyzed system.*

7 Examination and Evaluation of Candidate Projects

To give an example on how to apply the tailored method presented in section 5 a case from a Norwegian company considering adoption of FOS GIS will be examined. As seen in section 6 the company currently uses a web-based GIS, based on ESRI components, in combination with a proprietary GIS. The nature of this setup means that it could be replaced by a desktop GIS. As this study focus on FOS desktop GIS this case is ideal for an example evaluation.

The tailored method, shown in figure 5.2, will be applied and all relevant steps will be described in detail. The starting point is the interview conducted in section 6. The intention of this example is twofold: to give an example on how to apply the method on a given case and to present an overview of FOS desktop GIS. The main focus lies on the execution of the method. The result of this evaluation is a ranked list of one or more FOS desktop GISes that are likely to fulfill the requirements for the case in question, this is presented in section 8.

7.1 Usage Scenarios and Requirements

The first step is to determine what usage scenarios that apply to the case and extract the relevant requirements from these. This step requires an understanding of the case in question and an insight in the reasons the company has to consider the use of FOSS. An interview with three representatives from the company was carried out in order to shed light on these topics (see section 6).

As part of the interview a modified list of the 11 usage scenarios suggested by Cruz et al. (2006) was presented and the representatives was asked to consider how the different scenarios matched their envisioned use of FOSS. The results is presented in table 6.1. The most relevant usage scenarios where:

Scenario 2, Use FOSS with a long-time consideration.

Scenario 3, Use FOSS as a cost reduction model.

Scenario 8, Use FOSS for becoming independent of proprietary solutions and providers.

These three scenarios provides the starting point for the evaluation. Each scenario is linked to several requirements, and a requirement can be linked to more than one scenario. Using the relationship between scenarios and requirements, in addition to the insight in the companys thoughts on FOSS usage, a list of requirements for the case in question was compiled, as seen in table 7.1. One requirement proposed by the usage scenarios where removed, requirement 5.8 *Division of development*

Functional:	Technical:
Required functionality covered (3,8)	Target platforms supported (2)
Clear direction of product evolution recognizable	Reliability (8)
	Maintainability (2)
Political:	Organizational:
Possibility for influencing further development with respect to individual needs	Community exists (2,8)
Decrease of proprietary dependencies (8)	Sufficient support available (8)
	Long life existence (2)
Economical:	
Sustainability	
Protection of investment (3)	
Flexible maintenance according to individual needs (8)	
Cost reduction (3)	

Table 7.1: *Grouped requirements for the wanted FOS GIS. Requirements applying to scenario 2 marked (2), to scenario 3 marked (3), and to scenario 8 marked (8), requirements applying to all are unmarked.*

cost, on the grounds that there are no intention of doing FOSS development in the company.

This leaves 14 unique requirements in five categories to be used as a basis for the evaluation. Some of the requirements will influence the result more than others, as they apply to several of the scenarios or the interview indicated that they were important. Because of the emphasis of the human assessment in this evaluation no weighing of the requirements will be made, but the importance of each requirement is considered when making the final selection.

To evaluate how well the candidate projects fulfill the requirements a further decomposition is made. Each requirement is analyzed and a set of criteria is formulated for each. These criteria can be measured and rated, and when all criteria that make up a requirement is combined a score for the requirement is obtained. Some of the criteria used are taken from the frameworks presented in section 5, while others are made specifically for this evaluation. All the criteria can be rated from 0 to 2, and a description of what justifies a particular score is given. A criterion can apply to several requirements, just as a requirement can apply to several usage scenarios.

A complete overview of each requirement and the applicable criteria, in addition to guidelines for applying scores, is available in appendix B. A list of the 27 unique criteria used is presented in appendix C. In order to explain the reasoning used when determining the criteria the 14 requirements will be analyzed. The analysis is based on the description by Cruz et al. (2006). This analysis may also lead to the removal of some of the requirements, as a more thorough understanding of the requirement may reveal that it is not applicable. In addition to describing what criteria applies to the requirements a brief description of how to get hold of measurable data for each criterion is presented.

7.1.1 Functional Requirements

Functional requirements are closely related to the case in question, they are also the most important criteria. If the functional requirements are not met the product will not be of any use, no matter how well it scores on other requirements. The functional criteria are also the ones that are easiest to determine, usually a description page is available for the product. The relevant functional requirements are *required functionality covered* and *clear direction of product evolution recognizable*.

The interview revealed that the system is mainly used for visual inspection and some measuring and printing. In addition the system deals with different data formats, mainly spatial databases (ArcSDE) and WMSes. Support for shapefiles is required at the moment, but not in the future. In addition a “novice-friendly” GUI is wanted, as the operators tend to understand specific procedures rather than the underlying concepts. When expressed in a tabular form the required functionality

can be expressed as seen in table 7.2. In order to translate these criteria to a rating of 0, 1 or 2 a mapping is made. If all (*essential* and *wanted*) functionality are present a score of 2 is given, if all *essential* functionality are present a score of 1 is given, else a score of 0 is given. This is illustrated in table B.1.

To investigate how the different candidates fulfill these requirements the description pages for each product, as well as the tables provided by Steiniger and Bocher (2008)¹⁹ will be examined. In some cases where no information is found a test installation will be made.

Functionality	Importance
Map display with pan and zoom	Essential
Turning on/off layers	Essential
Measurements	Essential
Printing support (defined areas and resolutions)	Essential
Support Spatial Databases (ArcSDE)	Essential
Support WMS	Essential
“Easy” Graphical User Interface	Essential
Changing the cartography	Wanted
Support Shapefiles	Wanted

Table 7.2: *Required functionality for the system. Rated as Essential and Wanted, where essential is absolutely needed and wanted is “good to have”.*

A project should present clear thoughts on what direction the evolution is headed. This is essential in order to assure that the product will fulfill the functional requirements in the future. Usually FOSS projects have a “roadmap” or a timeline on their website describing future thoughts. Such matters can also be discussed on mailing lists or forums. The direction in which the product evolves should match the intended use for a particular case. The criterion used to determine whether a project has a clear direction of evolution is how well (if at all) roadmaps and timelines match the intended usage scenarios for the case in question, as seen in table B.2.

7.1.2 Technical Requirements

The relevant technical requirements are *target platforms supported*, *reliability* and *maintainability*.

Target platforms can be operating systems, libraries and virtual environments such as the Java platform. However, the relevant criterion here is operating system. The software must be available for the operating system used in the company, as a

¹⁹Provided at <http://spatialserver.net/osgis/>.

switch of operating system is a time consuming and costly project. The company in question uses MS Windows XP. Thus the criterion is that the candidate product must be able to run (natively) on Windows XP, and it is preferred that it runs on all three major OSes²⁰ as described in table B.3. Operating systems supported is usually listed on the download page.

Reliability is an important factor. The system should not crash unexpectedly and it should be robust. Factors that reveal how reliable the system is are bug statistics, age, how often patches are released, known problems in the past and the handling of these and whether the product is a *fork* or not. The criteria used to measure reliability are the two bug related criteria from the *Quality* category of the BRR framework (OpenBRR, 2005, p.18) and the *Maturity* criteria proposed by the QSOS framework (Atos Origin, 2006, p.16). This leads to the set of criteria listed in table B.4. In order to investigate these criteria the bug tracking systems of the various projects will be examined (where available) to gather data, in addition to general information provided at the project webpage.

Maintainability is defined as “how easy the system can be modified to correct faults, improve performance, or other attributes, or adapt to a changed environment” (Cruz et al., 2006). The QSOS framework features a set of maintainability criteria (p. 26, reproduced in table B.5) that will be used here. These criteria are concerned with source code and technical documentation, and thus inspection of source code (through svn repositories where available) or use of webpages presenting code statistics (such as ohloh.net)²¹ is required. Available developer documentation is another source of information.

7.1.3 Organizational Requirements

The organizational requirements are concerned with the community that develops and maintains a FOSS project. The relevant requirements identified are *the existence of a community*, *availability of sufficient support* and whether the community will have a *long life existence*.

An existing and active community is important. Without this the project probably has no future, which means that no new versions or bugfixes will be made. To measure how active the community is several indicators can be used, such as volume of mailing lists, number of code contributors, whether there are clearly identified developers, and activity on fixing bugs, developing new functionality and planning releases. The BRR framework provides criteria for determining mailing list activity and code contributors (*community* criteria, p. 19), combined with the *activity* criteria from the QSOS framework (p. 18) these will form the *community criteria* for this evaluation, as shown in table B.6. Data to determine the fulfillment

²⁰Windows, GNU/Linux and OS X

²¹See appendix H for an overview on calculation of code statistics.

of these criteria can be found in mailing list archives (specified on the identity card), code statistics pages and software development management systems (such as Sourceforge and Codehaus), where available.

Sufficient support should be available for the product, both in the form of personal support and documentation. Most FOSS projects use mailing lists to handle support requests and usually provides on-line documentation for the product. In addition there are in some cases available commercial pay-for-support solutions from third parties. To investigate the *support* requirement activity and climate on mailing lists, available documentation and the availability of professional support will be examined. These criteria are adopted from the *documentation* criterion of QSOS (p. 19) and the *support* criteria of BRR (p. 19), as seen in table B.7. Investigable data are found browsing mailing list archives²² and reviewing available user documentation found on project webpages. Some projects provide information on professional support on their webpage, in addition OSGeo maintains a list over companies offering professional support for FOS GIS products²³.

While an existing community is important it is also crucial that the community will continue to exist and function in the future. A project with an unstable future causes uncertainty in long time planning. The size of the community is a good indicator of whether a project can be expected to remain active, involvement of well reputed companies is also a good indication. Thus mailing list volume and project sponsorship (the QSOS *sponsor* criterion, p. 25) forms the *longlivity* criteria, as seen in table B.8. Data about sponsors are typically found on the project webpage, in some cases the community may be contacted to clarify matters.

7.1.4 Economical Requirements

Economical requirements are requirements that determine whether it is economically justifiable to adopt a FOSS product. In this case the relevant requirements are *sustainability*, *protection of investment*, *flexible maintenance according to individual needs* and *cost reduction*.

In order to protect investments there have to be stable development circumstances (i.e. the *reliability* requirements have to be fulfilled) and the migration to, and running of, the FOSS product must be expected to remain economically justifiable. This assessment is however beyond the scope of this study, and the company used in the case has not made any calculations on this. A candidate project can therefore be said to protect investments if it fulfills the *reliability* requirements, this requirement is thus disregarded.

For a project to offer flexible maintenance according to individual needs an important prerequisite is that the *required functionality* is covered. In addition

²²See appendix I for detailed mailing list statistics.

²³http://www.osgeo.org/search_profile

the development team needs to have sufficient skills and resources and the history of the project should show that open feature requests are closed within an acceptable timeframe. The criteria used to measure this are the *required functionality* criterion, the *professionalism* criteria of BRR (p.20) and an *open feature request fulfillment time* criterion, as seen in table B.9. Data can be found on project webpages and in the software development management systems, as well as through inquiries to the development team.

In order for a FOSS product to help a company reduce costs several requirements have to be met. If the required functionality is covered and the development moves in a wanted direction (ref: *maintenance according to individual needs* and *influencing further development*) a good basis is laid. The cost of migration is also important, but as stated earlier this lies beyond the scope of this study. Finally a minimum of product dependencies and a cooperative discussion forum is helpful. The criteria for the cost reduction requirement are thus *required functionality coverage*, *number (and type of) of dependencies* and the general *climate in support forums*, as seen in table B.10. Information on dependencies is in some cases listed on the project webpage, and for some architectures it is evident from the packaging of the release. In addition webpages such as <http://freshmeat.net> lists dependencies for several FOSS projects.

The sustainability requirement is, for the purpose of this evaluation, rather similar to the longlivity requirement, and is thus disregarded.

7.1.5 Political Requirements

The political requirements are crucial, as both the FOSS communities and the company adopting the FOSS are dependent on the rest of the IT world and political changes have side effects. The relevant political criteria are *possibility for influencing further development with respect to individual needs* and *decrease of proprietary dependencies*.

In order measure the possibility of influencing further development with respect to individual needs there is a need to examine several criteria. First of all, a good basis is laid if the project evolves in a satisfactory direction and have sufficient development resources. In addition a minimum of dependencies is wanted, and the community should be open to feature requests. The project should also have a history of efficient fulfillment of such feature requests. The criteria for this requirement is thus; the product should have a *clear direction of product evolution*, it should have sufficient resources (the *sponsor* criteria from QSOS, p. 25), the number of *dependencies* should be low, the mailing lists should have a *cooperative climate* and the number of unfulfilled *feature requests* over time should be low. These criteria are listed in table B.11. The criteria used here have been discussed earlier, thus methods for obtaining data are not discussed.

To make sure a shift to FOSS products helps decrease proprietary dependencies it is important that the *feature requirements* are met and that the project *evolves in the wanted direction*. In addition the presence of enough development resources and involvement of well-reputed companies are positive (the *sponsor criterion*). The criteria to determine ability to decrease proprietary dependencies is listed in table B.12. All these criteria are previously described along with methods for obtaining data.

7.1.6 Remaining Requirements and Criteria

The removal of two requirements means that the requirements table (table 7.1) is changed. The final set of requirements is presented in table 7.3. A total of 27 unique criteria will be used to determine whether these 12 requirements are fulfilled, these criteria are listed in appendix C.

Functional:	Technical:
Required functionality covered (3,8)	Target platforms supported (2)
Clear direction of product evolution recognizable	Reliability (8)
	Maintainability (2)
Political:	Organizational:
Possibility for influencing further development with respect to individual needs	Community exists (2,8)
Decrease of proprietary dependencies (8)	Sufficient support available (8)
	Long life existence (2)
Economical:	
Flexible maintenance according to individual needs (8)	
Cost reduction (3)	

Table 7.3: Final set of requirements for the wanted FOS GIS. Requirements applying to scenario 2 marked (2), to scenario 3 marked (3), and to scenario 8 marked (8), requirements applying to all are unmarked.

7.2 Initial Filtering

The second step in the evaluation method is an initial filtering of candidate projects. This initial filter is based on the *functional requirements* presented in

table 7.2 and in order for a candidate project to pass the initial filter it has to achieve a score of 1 or 2. This means that it must fulfill at least all the *essential* functional requirements.

As stated earlier the ten FOS desktop GIS projects presented by Steiniger and Bocher (2008) will serve as a starting point for the evaluation. The ten projects are listed in table 7.4.

GRASS	Quantum GIS
uDig	gvSIG
SAGA	ILWIS
MapWindow GIS	OpenJUMP
KOSMO	OrbisGIS

Table 7.4: *Candidate FOS GIS projects, taken from Steiniger and Bocher (2008). See appendix G for links to project websites*

Table 7.5 gives an overview on how the ten candidate projects matches the functional requirements. The tables provided by Steiniger and Bocher, as well as information provided on the project webpages (see appendix G for URLs) were used to gather the required information. A test installation was made in order to examine whether the requirements were met in two cases (OrbisGIS and OpenJUMP). Question marks mean that no information was found, but as long as the project fails at at least one other essential criteria it would have been disregarded anyways, so there is no need to investigate this further. When it comes to ArcSDE and printing support there are some products using external plug-ins or libraries. These will pass this screening, but will be given a score of 1.

When the results are examined and all projects with a rating of 0 are removed there are three candidates left, with the following scores:

- uDIG (2)
- gvSIG (2)
- OpenJUMP (1)

These three projects will be further investigated in phase 3 of the evaluation. It should be noted that the seven projects removed here are removed because they failed to meet the *particular requirements* for this case at *a given time*. All software are continuously evolving and different cases have different requirements. The fact that these projects are filtered out in this evaluation does not mean that they are unusable in other cases. The only way to tell is to apply an evaluation method such as the one described here.

Map display with pan and zoom	GRASS Yes	QGIS Yes	rdig Yes	gvSIG Yes	SAGA Yes	IWIS Yes	MapWindow Yes	OpenJUMP Yes	KOSMO Yes	OrbisGIS Yes
Turning on/off layers	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Measurements	Yes	Yes	Yes	Yes	Yes	?	Yes	Yes	?	?
Printing support (defined areas and resolutions)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Plug-in	Yes	?
Support Spatial Databases (ArcSDE)	OGR	No	Yes	Yes	No*	No	No	Plug-in	No*	No
Support WMS	Yes	Yes	Yes	Yes	?	No*	Plug-in	Yes	Yes	?
"Easy" Graphical User Interface	No	Yes	Yes	Yes	Yes	?	Yes	Yes	Yes	Yes
Changing the cartography	Yes	Yes	SLD	Yes	Yes	?	Yes	Limited	SLD	?
Support Shapefiles	OGR	OGR	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table 7.5: *Functionality of the candidate projects. *means under development, Plug-in means via a plug-in, OGR means via the OGR library and SLD means SLD support.*

7.3 Data Collection

The third step of the evaluation method is the gathering of information needed to populate the *identity card* and *evaluation sheet*. Templates for these can be found in appendix D (tables D.1 and D.2). The information provided in these tables will form the basis for the final selection.

The identity card is adopted from the QSOS framework (Atos Origin, 2006) and the evaluation sheet consists of the 27 unique criteria determined earlier. Some of the criteria for the evaluation sheet are dependant on subjective judgment (e.g. quality of code), while others are of a more directly measureable nature (e.g. age and bug statistics). In some cases data may be unavailable or subject to estimation. In addition all candidate projects are continuously evolving, thus the scores for the various criteria may change over time. Due to these limitations other similar evaluations may not produce the same results, and the results of this evaluation should be read with this in mind. The goal is none the less to make an as accurate and up to date evaluation as possible. In cases where statistics for the last six months are needed the period June–November 2008 serves as a reference.

This section will describe the three candidate projects in detail, and each of the 27 criteria will be discussed for each candidate. This is done in order to describe the employment of the described methods for gathering data, as well as to give a more thorough overview of the candidate projects. The identity card and evaluation sheet for each candidate is presented immediately after the examination of each project. (Tables 7.6 and 7.7, 7.8 and 7.9, and 7.10 and 7.11 respectively).

7.3.1 uDig (User-Friendly Desktop Internet GIS)

uDig was initiated by the Canadian company Refractions Research Inc. in 2004–2005, and is now in version 1.1.1. It is based on the Eclipse development platform, is written in Java and licensed under the LGPL. One of its main focuses are direct editing of databases and data over the Internet. Because of this the user base is steadily growing. Two forks are known, DivaGIS and jGrass, these focus on different aspects. Development is joined among the projects where possible.

uDig fulfills both the essential and wanted functional requirements for this case, as seen in table 7.5. It is written in Java and thus available on all major OSes. A roadmap is available through Codehaus as a list of feature requests. No written and coherent thoughts on development direction were found. The nature of the Codehaus system makes uDig open to feature requests. The requests seem to focus on improving existent features and the approved requests seem to match the usage scenario for the case in question. This gives a score of 2. Bug tracking is available through Codehaus, the statistics here shows that 69 issues (or bugs) were filed the last six months, of these are 17 resolved, which gives a ratio of fixed vs. opened of 25%. The Codehaus archives²⁴ dates back to February 2007 and shows eight versions prior to the current 1.1.1 release. These are mainly release candidates for version 1.1.0, which can be considered a stable production release. Version 1.1.1 is a bug fix release, this gives a score of 1 on stability. The current version lists one open bug rated “blocker”, that has been open since January 2007. Overall there are 236 issues rated Major, Critical or Blocker. This gives a score of 1 on known problems.

The program is mainly written in Java, some minor parts are written in other languages such as Groovy, Perl and JavaScript (< 1% in total). The source code is commented and readable, and an applied coding policy seems to be enforced. The complexity of the code is low, but domain-knowledge is required for several parts of the code. Documentation for developers are available, both general Eclipse documentation and uDig-specific guides, tutorials and code examples. The technical documentation seems to be up to date. When it comes to dependencies uDig relies on GeoTools²⁵, a Java code library which provides standards compliant methods for the manipulation of geospatial data (GeoTools, 2008). However there is a close cooperation between the uDig and GeoTools developers, as several developers are contributing to both projects. In addition uDig is dependent on the Eclipse platform. This is a well-supported and stable project with heavy industry backing and the chances for discontinuation of the Eclipse project are minimal. This justifies a score of 2 on dependencies.

²⁴<http://jira.codehaus.org/browse/UDIG?report=com.atlassian.jira.plugin.system.project:versions-panel&subset=-1>

²⁵<http://freshmeat.net/depends/download-all/53834/>

When it comes to mailing lists, uDig provides both an User and a Developer list. The user mailing list had an average of 39 posts/month the last six months, while the developer list is used more frequently, with 353 messages/month in average. For the sake of this evaluation the user list is the interesting one, leaving uDig with a score of 0. According to code statistics there were 16 unique code contributors the last six months.

In the last year 87 bugs were reported, of these a total of 31 are resolved or closed. This accounts to a medium activity on bugs. In the last year a total of 9 new feature requests have been reported. Of these only one is fixed, on same day that it was reported. This gives a low score on average time to fulfill open feature requests. The Codehaus tools provides functionality for reporting new feature and improvement requests. Combined there are made 30 such requests by a total of 13 different people over the last year. These requests, combined with bug reports are, linked to planned future releases. The activity on both functionality and releases must be considered good.

Commercial support for uDig is at least available from Refrations (uDig is not listed by OSGeo), which seems to offer both troubleshooting and installation support, as well as consulting and development services. The user documentation consists of a textual and video walkthrough, a FAQ page and an on-line manual (in English and Portuguese), managed as a wiki which seems to be up to date. Other than the separation between user and developer documentation there are no division to different target readers. Questions posted on the user mailing list seems to be answered quick, be written in a polite language and offer helpful advice and examples. This gives a score of 3.

uDig shares a large amount of code and contributors with GeoTools and related projects. About 5–6 people are working mainly on uDig, another 15–20 on various other projects related to GeoTools. The core group consists of about 6 developers. In order to get accepted into the core group one have to contribute accepted patches and be active in the community. The majority of the core group have been around for at least three years. This justifies a score of 2 on the *developers* criterion. uDig is supported by an independent foundation supported by several companies (Refrations, Camptocamp, Hydrologis, Lisasoft and Axios). Some of these companies are represented in the Project Steering Committee.²⁶

²⁶This paragraph is based on an IRC chat with Jesse Eichar, a main contributor to uDig (former Refrations employee).

General Information

Name	uDig (User-Friendly Desktop Internet GIS)
Description	Aims to provide a complete Java GIS solution. Focus on support for internet GIS standards (WMS, WFS), as well as spatial databases. Built on the Eclipse development platform.
Source	Sponsored by Refrations Research (commercial)
Licenses	uDig is LGPL, Eclipse is Eclipse Public License (EPL) (some modules GPL)
Website	http://udig.refrations.net/
Compatible OSes	MS Windows, GNU/Linux and Mac OS X
Fork origin	N/A

Functional and Technical Aspects

Technologies	Written in Java
Prerequisites	None
Detailed functionality	Data viewing, editing and analysis
Roadmap	Available at http://jira.codehaus.org/browse/UDIG?report=com.atlassian.jira.plugin.system.project:roadmap-panel
Bug tracking	Available at http://jira.codehaus.org/browse/UDIG?report=com.atlassian.jira.plugin.system.project:openissues-panel
Feature requests	Same url as bug track

Help and Support

Documentation	On-line user guide at http://udig.refrations.net/confluence/display/EN/Home
Free support	User mailing list at http://lists.refrations.net/pipermail/udig-users/ . Also channel on IRC (#udig on freenode)
Paid support	Available from Refrations Research (http://www.refrations.net/products/udig/support/).

Synthesis

General trend	Seems to focus on Web GIS standards and databases, possible to use as a PostGIS front-end.
Comments	Well-supported, seems determined on what direction to take and what to focus on. Sponsoring company involved in other FOS GIS projects as well. Relatively large user base.

Table 7.6: *Identity Card for uDig (User-Friendly Desktop Internet GIS).*

Criterion	Score		
	0	1	2
Required functionality present			✓
Clear direction of product evolution			✓
MS Windows Supported			✓
Number of open bugs for the last 6 months			✓
Number of bugs fixed the last 6 months	✓		
Age			✓
Stability		✓	
History, known problems		✓	
Forkability, source of forking			✓
Quality of source code			✓
Technological dispersion		✓	
Intrinsic complexity		✓	
Technical documentation			✓
Dependencies			✓
Average volume of mailing list last 6 months	✓		
Number of code contributors last 6 months		✓	
Developers, identification, turnover			✓
Activity on bugs		✓	
Activity on functionality			✓
Average time to fulfill open feature requests	✓		
Activity on releases			✓
Quality of professional support			✓
Documentation			✓
Support climate			✓
Sponsor			✓
Project Driver			✓
Difficulty to enter core developer team		✓	

Table 7.7: *uDig Evaluation Sheet*

7.3.2 gvSIG (Generalitat Valenciana, Sistema d'Informació Geogràfica)

gvSIG is a project initiated by the Regional Council for Infrastructures and Transportation in Valencia (Spain) with the goal of replacing ESRI's ArcView. Development was initiated in 2003 and is lead by the company IVER S.A., with several universities and other companies as contributors. gvSIG supports both vector and raster data, and has done a lot of work on database connectivity. The main programming language used is Java, thus allowing support for all the major OSes. The project is licensed under the GPL.

gvSIG fulfills all the functional requirements, as seen in table 7.5. A detailed roadmap is available for download at the project webpage, and gives an overview of *what* features planned to implement *when*. Focus seems to be on extended database connectivity and advanced data analysis. gvSIG has no public software development management systems like the other two candidates, but access to the internal bug and feature request system was granted. The last six months a total of 332 bug reports where filed, and 232 where fixed. This gives a ratio of fixed vs. opened of about 70%. The last six months a total of 13 feature requests where closed, at an average time of 1.5 months.

When it comes to stability there have been eight releases prior to the current 1.1.2 release since June 2004. According to the roadmap all the previous releases have introduced new functionality and are released with rather regular intervals (with the exception of the 1.1.1 release). Thus the project should be considered stable. According to available information there have been no major problems or crisis. The product is no fork itself and and the chances of forking are small.

The gvSIG source code is available through a read-only svn repository²⁷ (in addition to downloadable archives). It is readable and commented where needed, and it seems like an applied coding policy have been used. The majority of the code is Java, although some code are C++ (12%) and other languages such as C (5%) and Python (5%) This leads to a score of 1 on technological dispersion. The design and code seems to be easy to understand, but domain-knowledge may be needed for several modules. The project webpage has an own section with developer documentation, but most of the content is in Spanish. From an international viewpoint this accounts to a score of 0 on technical documentation. When it comes to dependencies, Steiniger and Bocher (2008) notes that gvSIG has dependencies to more than 100 Java and C++ libraries, on the gvSIG webpage seven of them are listed, these include GeoTools2, GDAL and JTS in addition to several math and general purpose libraries. The sheer number of dependencies gives a score of 0.

The gvSIG project hosts three mailing lists, a Developer and a User list in

²⁷http://groups.google.co.uk/group/onegislist/browse_thread/thread/27de663f86593426

Spanish and an international list in English. The international list (user and developer combined) had an average of 142 posts/month the last six months, the Spanish user and developer lists had averages of 311/month and 69/month respectively. In all cases this would have given a score of 1. When it comes to unique code contributors the last six months a total of 24 were registered, according to ohloh.net. It is however unclear how many of these can be considered developers. The project webpages maintains no list of developers, this gives a score of 1 on developer identification. There seems to be considerable activity on bug fixing, but the project maintains no public bug tracking system. Functionality requests seems to be driven by the core developer team, users submit feature requests through the mailing list. The roadmap shows that new functionality is planned for coming releases. The activity on releases seems to be top-notch, as noted .x versions are released at rather regular (6-month) intervals, in addition release candidates precede stable releases. The roadmap shows that releases are well-planned.

Professional (paid) support is not listed on the project website, but OSGeo lists seven companies providing support for gvSIG. These companies seem to offer full-fledged support. The user documentation is available as a downloadable document for each of the versions, in both English and Spanish. In addition online documentation is available, this is separated in a user-guide and installation guide. Both are available in English, Spanish, German, French and Italian. There is also a draft of the documentation for version 2.0 available (only in Spanish). Other than the distinction between user and developer documentation no further adoption to different target readers is made. The climate in the mailing list seems to be friendly, helpful and responses tend to be quick and to the point (these observations are based on the English mailing list). Thus all the support-related criteria qualifies for a score of 2.

gvSIG is, as mentioned, initiated and driven by the Regional Council for Infrastructures and Transportation in Valencia, which can be considered a project sponsor. In addition the European Union is funding the project (IT Business Edge, 2006). This qualifies for a score of 2 on sponsors. The project driver is also the council, which can be considered a “Group or corporation”, giving a score of 1. When it comes to the core group the project is controlled by the council, the criteria for joining the core group is not known, but it is likely that regular code contributors are not included in the core group and that the core group is selected based on political guidelines. This gives a score of 1.

General Information

Name	gvSIG (Generalitat Valenciana, Sistema d'Informació Geogràfica)
Description	A tool for management of geographic information. It is aimed at professionals and civil servants. Interface in 13 languages.
Source	Initiated by the Regional Council for Infrastructures and Transportation, Spain (governmental), developed by IVER Tecnològias (commercial)
Licenses	GNU Public License (GPL)
Website	http://www.gvsig.gva.es/
Compatible OSes	MS Windows, GNU/Linux and Mac OS X
Fork origin	Raster functionality from SAGA

Functional and Technical Aspects

Technologies	Written in Java
Prerequisites	Uses GDAL-OGR and GeoTools. JAI Image I/O and JAI required
Detailed functionality	Aims to replace ESRI ArcView (almost reached)
Roadmap	Available at http://www.gvsig.gva.es/index.php?id=1814&L=2
Bug tracking	Not public (web form planned)
Feature requests	Not public (web form planned)

Help and Support

Documentation	Available at https://gvsig.org/web/docusr
Free support	Mailing list at http://runas.cap.gva.es/mailman/listinfo/gvsig_internacional
Paid support	7 companies listed by OSGeo

Synthesis

General trend	Seems to be focusing on Web GIS standards and databases.
Comments	Not very open in terms of bug tracking and source code availability, international community and rather complete software

Table 7.8: *Identity Card for gvSIG (Generalitat Valenciana, Sistema d'Informació Geogràfica).*

Criterion	Score		
	0	1	2
Required functionality present			✓
Clear direction of product evolution			✓
MS Windows Supported			✓
Number of open bugs for the last 6 months		✓	
Number of bugs fixed the last 6 months			✓
Age			✓
Stability			✓
History, known problems		✓	
Forkability, source of forking			✓
Quality of source code			✓
Technological dispersion		✓	
Intrinsic complexity		✓	
Technical documentation	✓		
Dependencies	✓		
Average volume of mailing list last 6 months		✓	
Number of code contributors last 6 months		✓	
Developers, identification, turnover		✓	
Activity on bugs		✓	
Activity on functionality		✓	
Average time to fulfill open feature requests			✓
Activity on releases			✓
Quality of professional support			✓
Documentation			✓
Support climate			✓
Sponsor			✓
Project Driver		✓	
Difficulty to enter core developer team		✓	

Table 7.9: *gvSIG Evaluation Sheet*

7.3.3 OpenJUMP (Open Java Unified Mapping Platform)

OpenJUMP was initiated by the JPP Development Committee as an attempt to merge the different forks of the original JUMP that was founded in 2002 by a consortium of Canadian companies and provincial ministries. The original JUMP project stopped in 2004 due to loss of financial support. OpenJUMP is written in Java and released under the GPL, with some LGPL components. The original objective was to develop a GIS specifically for data editing and conflation, this causes the project to focus on vector data. Other JUMP forks include DeeJUMP, SkyJUMP and PirolJUMP.

OpenJUMP fulfills most of the functional requirements, although printing and ArcSDE access requires a plug-in, as seen in table 7.5, which gives a score of 1. The software is available for all three major OSes, as it is written in Java. Several roadmap documents are available on the webpage, describing thoughts about the future development as well as well as to-do lists for the next release. No roadmap is available through the Sourceforge pages, but bug reports, feature and supports requests can be tracked there. Three bugs where opened the last six months, one of them have been closed. This gives a ratio of fixed vs. opened bugs of 33%. The download page shows 5 releases, version 1.0 with an additional maintenance release and version 1.2 with two beta-releases. The change log between the two latest versions lists 64 changes, of which 40 are new functionality. This justifies a stability score of 2. The history shows no major problems or crisis.

The source code is readable and commented where needed. A coding policy seems to be applied, and the code does not seem very complex, although domain-expertice may be needed in some cases. The program is written entirely in Java. The developer documentation consists of a downloadable developer guide for the original JUMP project, in addition to a developer FAQ and a wiki containing notes, how-tos and tips & tricks. The wiki is not complete, nor especially up to date. The dependencies of OpenJUMP are not listed on the project webpage, neither is OpenJUMP listed on freshmeat.net. However, it is clear that OpenJUMP is dependent on JTS²⁸. In addition the batik and log4j libraries and several Java math libraries are used.

OpenJUMP provides a user mailing list in addition to a developer mailing list through Sourceforge. Both these lists are intended for all the JUMP forks. The user list had an average of 59 posts/month the last six months, while the developer list had an average of 130/month. In the last six months a total of five unique contributors submitted code to the project²⁹. The website lists about 20 people involved in developing the system, but in general there are 4-6 core programmers and an additional 2-3 persons who regularly contribute non-programming work.

²⁸The Java Topology Suite (see Vivid Solutions, 2008).

²⁹According to <https://www.ohloh.net/p/9819>.

This gives a score of 2, as the team seems to be stable and committed to the task. The bug tracker lists a total of 118 reported bugs, of these are 30 still open, the oldest have been open since 2005. 28 of the bugs are older than six months, 27 of the open bugs are not assigned to a specific developer. This gives a rating of 1. A total of 89 feature requests are listed on the Sourceforge pages, 48 of them still open. Of the open requests the oldest dates back to 2005, and only one is newer than six months. Most of the requests come from a total of about 10 people. While many feature requests are still open, there seems to be interaction with the planned direction of the project. This, combined with the use of a tool for managing such requests, justifies a score of 2. In the last year 14 feature requests were reported, of these have 6 been closed, at an average of 3.5 months. When it comes to releases and planning of these the project provides a wiki page describing tasks to be completed before the release of the next version. There are nightly builds available, and the history shows that there are usually one or two beta-releases before a .x release. This qualifies for a score of 2.

The project webpage provides links to 12 companies worldwide offering paid support and development, OSGeo lists an additional two companies. These companies seem to provide whatever support needed, which gives a score of 2. The user documentation includes the original JUMP documentation and OpenJUMP tutorials in several languages. A FAQ exists in wiki-form, but it is not updated in six months. An English user guide for the current version is available as a wiki, and seems to be fairly up to date (still in progress). This gives a score of 1. The user mailing list seems to be a friendly place, with helpful answers in a rather short time and follow-ups where needed. A score of 2 is given.

As stated earlier the project is led by the independent JPP Development Committee and developers from several commercial companies contribute code to the OpenJUMP project, while developing their own JUMP forks. As such there are no explicit sponsors of the project, but it benefits from being part of the JUMP family, which justifies a score of 2. This also means that there is a clear project driver present in the JPP committee with support from corporations. The core development group is rather informally arranged and consists of the most active developers with one person acting as a “benevolent dictator”. Major strategic decisions are usually dealt with through a voting process on the developer mailing-list. This gives a score of 1 on difficulty to enter core group.

General Information

Name	OpenJUMP (Open Java Unified Mapping Platform)
Description	Strong focus on vector data creation and analysis. Focusing on merging forks.
Source	Originally Vivid Solutions (commercial), now JUMP-Pilot Project (community)
Licenses	GNU Public License (GPL), some LGPL components
Website	http://openjump.org/wiki/show/HomePage
Compatible OSes	MS Windows, GNU/Linux and Mac OS X
Fork origin	JUMP

Functional and Technical Aspects

Technologies	Written in Java
Prerequisites	None
Detailed functionality	(Vector) editing and analysis
Roadmap	http://openjump.org/wiki/show/Some+Possible+Goals+For+OpenJUMP
Bug tracking	http://sourceforge.net/tracker/?group_id=118054&atid=679906
Feature requests	http://openjump.org/wiki/show/FeatureRequests and http://sourceforge.net/tracker/?group_id=118054&atid=679909

Help and Support

Documentation	http://openjump.org/wiki/show/Documentation
Free support	http://groups.google.com/group/openjump-users?pli=1
Paid support	7 companies listed by OSGeo, see also http://openjump.org/wiki/show/Professional+Support

Synthesis

General trend	Focus on plug-ins and light core, merging of forks
Comments	Open on code, bugs and requests small core developer group, supported by commercial developers

Table 7.10: *Identity Card for OpenJUMP.*

Criterion	Score		
	0	1	2
Required functionality present		✓	
Clear direction of product evolution			✓
MS Windows Supported			✓
Number of open bugs for the last 6 months			✓
Number of bugs fixed the last 6 months		✓	
Age			✓
Stability			✓
History, known problems		✓	
Forkability, source of forking		✓	
Quality of source code			✓
Technological dispersion			✓
Intrinsic complexity			✓
Technical documentation		✓	
Dependencies		✓	
Average volume of mailing list last 6 months		✓	
Number of code contributors last 6 months	✓		
Developers, identification, turnover			✓
Activity on bugs		✓	
Activity on functionality			✓
Average time to fulfill open feature requests		✓	
Activity on releases		✓	
Quality of professional support			✓
Documentation		✓	
Support climate			✓
Sponsor			✓
Project Driver			✓
Difficulty to enter core developer team		✓	

Table 7.11: *OpenJUMP Evaluation Sheet*

7.4 Ranking and Selection

The final part of the evaluation is the manual selection and rating. In order to get an overview of how the candidate project scored on each of the twelve requirements the evaluation sheets were inspected. The average score of all the criteria that make up the requirement was computed, thus leaving a ranking for each of the candidate projects, showing how well it fulfills each requirement. All requirements are ranked between 0 and 2, where 2 is the best. A score of 1 or lower indicates that the candidate will have problems fulfilling the requirement.

Table 7.12 shows how the candidate projects scored on each of the twelve requirements, a graph showing these scores is provided in figure 7.1. Appendix E shows the tables used to calculate these scores.

		uDig	gvSIG	OpenJUMP
Functional	Required functionality	2.00	2.00	1.00
	Evolution direction	2.00	2.00	2.00
Technical	Target platform	2.00	2.00	2.00
	Reliability	1.33	1.67	1.50
	Maintainability	1.50	1.00	1.75
Organizational	Community	1.33	1.17	1.17
	Support	1.50	1.75	1.50
	Longlivety	1.00	1.50	1.50
Economical	Flexible Maintenance	1.25	1.50	1.25
	Cost reduction potential	2.00	1.33	1.33
Political	Influence of development	1.60	1.50	1.75
	Decrease of proprietary dependencies	2.00	2.00	1.67

Table 7.12: Scores for each of the 12 requirements used in the evaluation. All scores are calculated as the average of the scores on the criteria that makes up the requirement.

A quick inspection of the scores show that none of the candidate projects scored less than 1 on any of the requirements. This means that all three products are usable for the case in question. There are however some differences, and these are discussed in the following, grouped by category. The final selection and recommendation are presented in section 8 together with a discussion of the results.

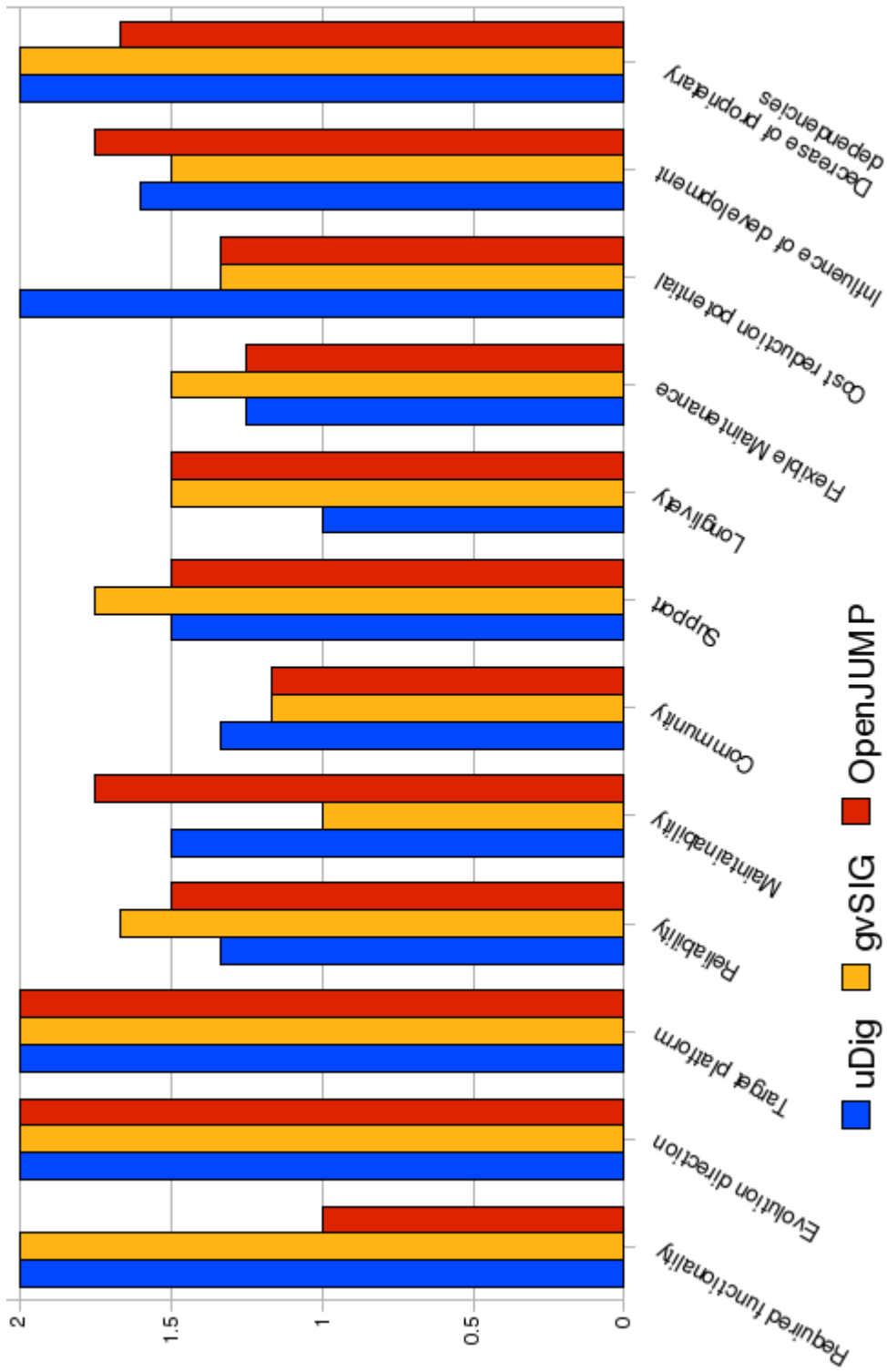


Figure 7.1: Graph showing the scores presented in table 7.12.

7.4.1 Functional Requirements

The functional requirements are *required functionality* and *evolution direction*. The required functionality depends on the fulfillment of the requirements presented in table 7.2. Table 7.5 shows that both uDig and gvSIG met both all the essential and wanted requirements, while OpenJUMP got a score of 1 due to the use of plug-ins for ArcSDE and printer support, in addition to the fact that changing of cartography where reported to be limited.

Evolution direction describes how well the planned evolution direction of the project matches the usage scenario. All three candidates got a score of 2 on this requirement. This means that uDig and gvSIG are the candidates ranking highest on this requirement. As noted in section 9 OpenJUMP's performance on the functional issues should be investigated further before drawing any conclusions.

7.4.2 Technical Requirements

The technical requirements are *target platform*, *reliability* and *maintainability*. All three candidate projects are written in Java, and are thus available on all the major OSes. The reliability requirements are dependent on bug statistics, age, stability and history and whether the project is a fork or are likely to be forked. All three candidates obtained a score of 2 on target platform. On the reliability requirement gvSIG got the highest score, followed by OpenJUMP and uDig.

OpenJUMP scored slightly higher than uDig on this requirement. If the bug statistics criteria are excluded gvSIG scores higher than the two other candidates (1.75 compared to 1.50 for uDig and OpenJUMP).

The maintainability requirement deals with source code and developer documentation. Although the quality of the source code is high for all projects, OpenJUMP is better when it comes to technological dispersion and intrinsic complexity. The fact that much of the developer documentation for gvSIG is available only in Spanish means that OpenJUMP must be considered the best candidate here, although uDig beats it on developer documentation.

Overall this means that OpenJUMP is considered the best candidate when looking at the technical requirements.

7.4.3 Organizational Requirements

The organizational requirements are *community*, *support* and *longevity*. On the community requirement uDig scored high on identification of developers and activity on releases, and functionality while the mailing list activity is low. OpenJUMP had the least number of contributors, and gvSIG scored rather average on most of the criteria. uDig got the highest overall score on this requirement.

On the support requirement all candidates scored rather high, uDig scored low on mailing list activity, while the OpenJUMP documentation is not as good as the others. This means that gvSIG scored highest on this requirement.

The low mailing list activity means that uDig lost points on the longlivity requirement as well, half a point behind the others. This means that gvSIG are the best candidate when looking at the organizational requirements. It should be noted that uDig is the only project that uses an IRC channel as a communication method, and this may be a reason for the low mailing list activity, as some requests may be handled here. If this activity accounts for a score of 1 on mailing list activity uDig would be considered the best candidate for the organizational requirements.

7.4.4 Economical Requirements

The economical requirements are *flexible maintenance* and *cost reduction potential*. gvSIG scored highest on the flexible maintenance criteria, with the two other candidates 0.25 points behind.

The cost reduction potential requirement is dependent on presence of required functionality, dependencies and support climate. uDig obtained a score of 2 on all these criteria, while gvSIG scored low on dependencies. OpenJUMP scored lower due to dependencies and functional requirements, meaning that uDig was best on this requirement.

As noted before the low score for OpenJUMP on functional requirements will have to be investigated further. Both the economical requirements are dependent on this criterion, and the results may thus be misleading. However, uDig scored highest on the economical requirements, and would have got a slightly better score than OpenJUMP even if the project got a score of 2 on the functional requirements. This means that the best candidate with respect to economical requirements is uDig.

7.4.5 Political Requirements

The political requirements are *influence of development* and *decrease of proprietary dependencies*. The two criteria determining the outcome where dependencies and average time to fulfill open feature requests, as the candidates got the same scores on the other criteria. This means that OpenJUMP got a slightly better score than uDig and gvSIG.

The decrease of proprietary dependencies requirement was generally scored high on, with both uDig and gvSIG obtaining a score of 2. OpenJUMP lost some points due to the low score on functional requirements. If OpenJUMP proves to fulfill this criterion the scores will be equal. As the standings are now uDig and gvSIG are best on this requirement.

This means that uDig are the best candidate when looking at the political requirements, although OpenJUMP would beat it with a full score on the functional requirements.

8 Recommendations

The purpose of this section is to analyze the results of the evaluation and make a suggestion for what candidate product to use in the case analyzed in this study. It is important to be aware that the results of this evaluation applies to a specific case, and thus the final recommendations can not be treated as a generic ranking of the “best” FOS desktop GIS.

As emphasized earlier the final selection is dependent on human assessment, and the scores obtained and discussed in section 7 (presented in table 7.12) should be treated as guidelines only.

8.1 Analysis

At the end of the examination the different requirements where inspected and a “winning” candidate where identified for each. The selection of a overall “winning” candidate is although not as easy as picking the candidate “winning” the most categories. It is important to remember that the twelve requirements comes from the three relevant usage scenarios defined for the case. Some of the requirements apply to several usage scenarios, and should thus be weighed higher. These are:

- Evolution direction - all scenarios

- Influence of development - all scenarios

- Required functionality - scenario 3 and 8

- Community - scenario 2 and 8

These requirements should be considered the most important ones, with the functional requirement as the most important one due to its nature.

Both uDig and gvSIG scored 2 on required functionality, while OpenJUMP got 1 point. This should suggest that OpenJUMP should be ruled out, but as discussed earlier, an actual installation test (described in section 9) should be carried out in order to investigate this matter further.

All three candidates scored 2 on evolution direction, this means that this requirement makes no distinction between the candidates. OpenJUMP scores slightly higher on influence on development, while uDig scores higher than the two others on community.

gvSig got the highest score on reliability, support and flexible maintenance, OpenJUMP on maintainability and influence of development, while uDig scored highest on community and cost reduction potential. On the other requirements there where two or three candidates sharing the top spot. This means that there are no obvious winner, as the different candidates excels on different requirements. The

requirements with the biggest difference in scores (neglecting functional requirements) are maintainability, longlivity and influence of development. gvSIG scored lower than the others on maintainability, while uDig scored lower on longlivity and higher on influence of development than the others.

OpenJUMP where considered the best candidate in the technical category. gvSIG where the best in the organizational but if the criterion regarding mailing list activity had included IRC activity it is possible that it could have been topped by uDig. uDig ranks highest in both the economical and political categories.

Another point worth noting is that uDig is the only candidate with a score of 2 on cost reduction potential, while the other candidates scores below 1.5. This is one of the biggest differences in this evaluation. As one of the main reasons for considering FOSS adoption in this case where cost reduction, this requirement should be emphasized.

8.2 Final Recommendation

If these results are compared there are no clear winner, but in general uDig performs well. The only requirements where it got the lowest score where reliability and longlivity. On the case of cost reduction it is the clear winner, and the investigation in section 7 found that it was the best candidate in several of the categories.

This means that this study concludes with the recommendation of *uDig*, with the note that the community aspect should be investigated further. In addition a test installation should be carried out and a thourough test case applied in order to ensure that the required functionality are actually present and works as it should. Some thoughts on this are presented in section 9.

If further tests should reveal that uDig is not capable of fulfilling the functional requirements the other two candidates should be examined. They both performed well, and there are in general small differences on the scores they obtained in this evaluation. This means that all the three candidates fulfill the requirements in the case investigated, but there are some small differences that makes it possible to select one of them.

9 Further Work

The evaluation method presented in section 5 and applied in section 7 is intended to produce a final set of FOSS products that suits the needs for a particular case, in a particular company. The method focuses on several aspects, such as economical, political and organizational, in addition to the functional and technical requirements that are usually considered the most important. It is noted that functional requirements are essential, in the sense that candidate projects that fail to meet the functional requirements are filtered out.

The evaluation of functional requirements in this study is rather limited, and to a large degree based on the findings of Steiniger and Bocher (2008). While their general overview and functionality tables are quite thorough and can be considered up to date³⁰, they can not cover all aspects of a product. The information provided on project webpages should be expected to be up to date and accurate, but this is not always the case. Features can be unstable or not tested a particular environment. The intended use can be complicated if the product isn't designed with it in mind.

As noted in section 8 a test installation should be carried out and a thorough test case applied in order to confirm whether the functional requirements are actually met. In the case of OpenJUMP, ArcSDE and printer support should be examined. This project got a lower score on functional requirements, as these features were reported to be available only through plug-ins. A function available as a plug-in usually means that the functionality is of lesser interest to the developing community, and quality and support may thus be limited. This may however not always be the case. Carrying out a test installation could reveal whether this is the case.

Another aspect that should be considered in more detail is the *usability*. In section 4 it was noted that usability traditionally have been an issue for FOSS products, although the later years have brought greater attention to usability in the FOSS communities. As with functional requirements the best way to examine the usability of software is by means of a test installation. Such a test installation was not carried out in this study, as it would have broadened the scope of the project even further. In order to investigate usability issues the users of the intended system would have to be involved, and considerable effort would have to be made to set up data sources, install and configure the software and defining test cases.

One advantage when evaluating FOSS products is that the software can be downloaded and installed at no cost. This means that a test installation only requires an investment in the man-hours needed to perform the evaluation, and no agreement with the vendor is required prior to testing. Because of this test instal-

³⁰Both tables were reviewed in March 2008.

lations are made cheaper and easier than with proprietary software. This aspect should be used as an advantage, thus should test installations be a mandatory task when evaluating FOSS. As a consequence a possible approach for the evaluation of *functional requirements* and *usability issues* through a test installation is outlined here, in order to provide pointers and guidelines for such an evaluation. As with the evaluation carried out it is important to follow a defined procedure in order to obtain usable metrics. Thoughts on how to set up such procedures are presented in the following.

9.1 Test Installation Guidelines

In order to check whether the required functional requirements are indeed met a test installation can be made. In advance features to be tested should be decided, and data to be used should be available. In this particular case this would mean that a WMS delivering spatial data and an ArcSDE database containing data should be set up, with an user account for the test installation. If possible a “clean” computer, matching the computers envisioned used for the case, both with regards to hardware and operating systems, should be used.

The actual installation should be carried out according to installation guidelines supplied at the project webpage (if any). Then the datasources should be set up according to available documentation. When the system is set up and seems to be working (i.e. no error messages appear and the data seems to be loaded) a more formal test should be carried out. Braude (2001, chapter 9) describes in detail the testing process for a system development process. The test installation case for testing FOSS can be thought of as *Installation Testing*, but with the limitation that only the required functionality is tested.

Some kind of standard to describe and document the test process should be used, such as the ANSI/IEEE standard for test documentation provided by Braude (2001, p. 453). The test installation procedure should at least include:

- A test plan (i.e. *what is to be done*)
- Test cases
- Test procedures
- A test log
- A summary of the test

Further work is needed in order to establish a set of guidelines and procedures for conducting such a test. The tests should be easy and rather inexpensive to

perform, and they should be able to reveal whether the reported features are available and usable. A method for analyzing the results and incorporate them in the evaluation method described here would be interesting.

9.2 Usability Test Guidelines

As noted by Nichols and Twidale (2002) usability design traditionally have been neglected in FOSS development, and Ahearn et al. (2006) found that the *client tier* of a FOS GIS stack was the least mature. As discussed lack of usability consideration may be a reason for this. An assessment of the usability of the candidate projects should therefore be conducted before a migration to such software.

Usability is defined by ISO (1998) as:

“The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.” (ISO 9241-11)

Note that usability is dependent on *specified users* in a *specified context*. The usability evaluation should therefore include the actual end-users of the system, and the test setup should resemble the actual usage scenarios for the product. Shneiderman and Plaisant (2005, chapter 4) covers the topic of usability testing in rather great detail and describes both *Expert Reviews* and *Usability Testing and Laboratories*. One method that can be applied to test the usability of a candidate FOSS product is a variant of *field testing*.

A field test would present an actual user for a test-installation of the product. The user would be asked to carry out a set of tasks the system should support. Observation of the user while he or she performs the tasks, and an interview session after the tasks are completed can reveal problems with the system. Techniques such as the *think aloud protocol* and *videofilming* should be applied in order to capture as much information as possible.

It should be noted that usability tests are not concerned with whether the functionality is present, but with how the functionality enables the user to carry out his tasks. This means that the availability of required functions will have to be assessed beforehand. An usability test is therefore likely to be carried out after a *test installation*. Only candidates that are considered viable options based on the evaluation described in this study, and has undergone a test installation, should be subject to usability testing, as this kinds of tests require more resources than a one-man test installation.

A set of guidelines for how to carry out such an usability test and how to document it should be created. In addition tasks and issues to be examined for the specified case will have to be investigated prior to conducting such a test.

10 Discussions and Conclusion

This section is an attempt to discuss the evaluation method presented and present some concluding remarks on the work carried out in this study. While the evaluation method presented is intended to function as a tool for FOS GIS evaluation and cover all relevant aspects, it still has some limitations. These will be discussed and possible enhancements of the method will be described.

An important aspect when applying such an evaluation method is the availability of the data used. The example execution presents some possible sources and methods for obtaining this data, but there is still room for improvements. This section will discuss how the FOSS community can help companies conducting such evaluations by providing up to date and accurate data about their project.

The final recommendation process is the most important part of such an evaluation and, since the proposed method relies on human assessment, the process of final ranking and selection will be discussed in order to identify possible pitfalls and ways to avoid them.

The study will be concluded with some thoughts on the overall evaluation process and the state of art in FOS GIS.

10.1 Limitations - Discussion

While an evaluation method can shed light on several important aspects it is important to recognize that it is not without limitations. As noted several of the criteria used are dependant on subjective judgment. This means that the person conducting the evaluation must strive to be as objective and thorough as possible. A way of ensuring that the right scores are assigned could be to let more people carry out the evaluation and individually rate the different criteria. The scores could then be compared, and the criteria where scores differ would be subject to discussion among the evaluators. This would ensure a more objective evaluation.

A problem with the presented evaluation method is the fact that it relies heavily on reports by others when it comes to assessment of functionality. The method could be improved by incorporating some kind of *test installation* and *usability evaluation*, as described in section 9. Carrying out these tasks were beyond the scope of this study, but should be considered when applying this method.

This study argues that using an evaluation method like the one proposed here ensures that a broader range of alternatives are evaluated. This aspect did however not play a big role in the example evaluation executed, as a predefined list of ten candidate projects were used as a starting point. This may have lead to the exclusion of some candidates. Using a predefined set of candidates means that no examples on how to gather candidate projects were given. This should have been done, as the task of selecting as many candidates as possible increases the

propability that the final recommendation will be the best fit.

In spite of these limitations the evaluation method has proved usable. It has provided an insight in the current state of FOS GIS, as well as come up with a recommendation for a FOS GIS for the case in question. The evaluation carried out are described in detail, and can serve as an example for other evaluations or inspiration for the creation of similar methods.

It should also be noted that the initial filtering removed 70% of the candidates, leaving three projects to be examined in detail. This number of candidates proved to be practical. The gathering of information needed to populate the identity cards and evaluation sheets for all ten candidates would have been time-consuming and demanded more resources. The initial filter step is thus an important aspect of such an evaluation.

10.2 Data Availability - Discussion

The way FOSS projects are organized ensures that much data are freely available. Most projects use some kind of *version control system* such as svn. This ensures that the source code and contributions can be inspected and tracked. Several projects also use some kind of software development management system such as Sourceforge, and these are often open to the public. These tools helps gathering statistics on bug fixing, feature requests and planned releases. In addition webpages such as ohloh.net and freshmeat.net index these tools and present more readily usable statistics and information. The mailing list archives for a project are usually available as well. This simplifies the task of examining the community.

Although these webpages and tools makes it easy to gather information there where still some problems. In many cases data had to be collected and aggregated manually in order to obtain statistics. An example is code contribution statistics, all contributors to the project had to be examined in order to check wether they had made a contribution in the last six months. In order to be even more transparent the projects should compile such statistics and present them on their webpage. The criteria used in this study could be used as guidelines for what kind of statistics to provide.

Roadmaps or timelines are usually supplied by FOSS projects, although the format and organization of these ranges from collections of bug reports and feature requests compiled more or less automatically, to written, plain text documents describing thoughts on evolution and traditional timelines. A written or manually compiled document should be available, as this is easier to assess and get an overview of.

The information that proved most difficult to gather where the more *tacit information* about the community and core developer team. This information is considered known to all the involved developers and contributors, but can be hard

to obtain for an outsider. As a consequence few projects provide specific information on such matters. In this case inquiries to the project mailing lists triggered useful replies. As with statistics the FOSS communities should realize that the use of evaluation methods relying on assessment of organizational matters are increasing as FOSS gains popularity. In order to promote the products, information on community organization, sponsors and the core group should be listed on the project web page and be maintained so that it is up to date and accurate. Another important aspect is that bug tracking and feature request handling system should be open to the public. In the case of gvSIG access to information had to be obtained by contacting the developer community. This complicated the evaluation task, and should not be necessary.

10.3 Final Selection - Discussion

The task of making a manual selection based on a set of scores may seem difficult, but this ensures that the evaluator considers what the scores *represent* rather than just a blind computation. There are however a chance that the rated scores may be used the wrong way and that focus moves away from human assessment. Another problem is that several of the requirements relies on the same criteria. This means that a low score on one criterion may cause a low score on several requirements. This was the case with OpenJUMP, a score of 2 on the functional requirement criterion would have changed its score on four of the requirements.

The selection part revealed that the scores where rather similar for some of the requirements. This made the task of selecting the best candidate more difficult, as the differences where minimal and difficult to spot. The lack of an actual test installation also proved to be a problem, as there where a large degree of uncertainty related to the fulfillment of the functional requirements. For other evaluations using this method a test installation as described in section 9 should be made for each of the remaining projects.

The final selection consisted of three candidate projects, this was an adequate number of candidates for the comparison. If five or more candidate projects had made it to the final selection the task would have been more difficult, as each of the candidates had to be compared to each other. A possible solution would have been to remove any candidate that where below a given threshold on any or some specified requirements. In the example evaluation carried out here a threshold of 1 where used, but none of the candidates where affected by this

10.4 Concluding Remarks

This concludes this study. The reader is encouraged to use the documents, sources, guidelines and thoughts presented here when evaluating FOSS products for use in

a commercial setting. Such an evaluation method can reduce the risk associated with FOSS adoption, and ensure that the FOSS experience is a positive one. If in doubt of whether FOSS are worth considering there are several organizations working to inform the public of FOSS usage, Friprogsenteret is an example in Norway. The OSGeo Foundation can also be a source for information on FOS GIS in particular. Another source of information are the communities developing the systems. As long the posed questions are written in a friendly and understandable manner most communities are happy to provide information about, and help with, their products.

The absence of license fees and the openness makes FOSS easy to explore and test. By applying the method presented here this available information can be analyzed and assessed in a controlled manner which should make a selection easier. This work is an attempt to describe an evaluation method that utilizes this available information in order to make the right choices. As more commercial firms consider adoption of FOSS products the need for methods such as this increases. It is believed that this study can serve as an eye opener and encourage more companies to adopt FOSS. This study shows that there are FOSS alternatives to most proprietary GIS packages available, and there are a large pool of candidates to choose from. The number of FOS GIS projects are likely to increase in the future, emphasizing the need for such evaluation methods.

References

- Ahearn, S., Almeida, D., and Gahegan, M. (2006). Proof of concept and state of the art in FOSS Geospatial Technology. Technical report, FIGS Working Group.
- Atos Origin (2006). Method for Qualification and Selection of Open Source software (QSOS), version 1.6. (Available online at http://www.qsos.org/?page_id=3).
- Barndt, M. (1998). Public Participation GIS – Barriers to Implementation. *Cartography and Geographic Information Science*, 25(2):105–112.
- Braude, E. J. (2001). *Software Engineering: An Object-Oriented Perspective*. John Wiley & Sons, Inc.
- Buchanan, T. R. (2005). Comparison of Geographic Information System Software (ArcGIS 9.0 AND GRASS 6.0): Implementation and Case Study. Master's thesis, Hardin-Simmons University, 2200 Hickory, Abilene, Texas, USA.
- Câmara, G. and Onsrud, H. (2003). Open Source GIS Software: Myths and Realities. In *International Symposium on Open Access and The Public Domain in Digital Data for Science, UNESCO, Paris*.
- Cox, A. B. and Gifford, F. (1997). An Overview to Geographic Information Systems. *The Journal of Academic Librarianship*, 23(6):449–461.
- Cruz, D., Wieland, T., and Ziegler, A. (2006). Evaluation criteria for free/open source software products based on project analysis. *Software Process Improvement and Practice*, 11(2):107–122.
- Dagens It (2008). Fjernet Microsoft fra kartet (Norwegian). (Available online at <http://www.dagensit.no/trender/article1383612.ece>, retrieved 10.10.2008).
- Egenhofer, M. J. and Kuhn, W. (1998). Beyond Desktop GIS. A Family of Portable Spatial Information Technologies. In *GIS Planet 1998 Annual Conference Proceedings*, Lisbon, Portugal.
- ESRI (2008). Online GIS Dictionary. (Available online at <http://support.esri.com/index.cfm?fa=knowledgebase.gisDictionary.gateway>).
- Feller, J., Fitzgerald, B., Lakhani, K. R., and Hissam, S. A., editors (2005). *Perspectives on Free and Open Source Software*. The MIT Press, Cambridge, Massachusetts.

- Fitzgerald, B. (2006). The transformation of open source software. *MIS Quarterly*, 30(3):587–598.
- GeoTools (2008). GeoTools Webpage. (Available online at <http://geotools.codehaus.org/>).
- Golden, B. (2005). Making Open Source Ready for the Enterprise, The Open Source Maturity Model. *Extracted from Succeeding with open source, Addison-Wesley (available online at <http://www.navicasoft.com/pages/OSMMwhitepaper.html>)*.
- Gray, J. (2008). Desktop GIS for Linux: An Introduction. *Linux Journal* (available online at <http://www.linuxjournal.com/content/desktop-gis-linux-introduction>).
- Hauge, Ø. (2008). Bruk av fri programvare i norsk programvareindustri (Norwegian). Technical report, NTNU.
- Heywood, I., Cornelius, S., and Carver, S. (2002). *An Introduction to Geographical Information Systems*. Pearson Education Limited, Essex, UK, 2. edition.
- ISO (1998). ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11: Guidance on usability.
- IT Business Edge (2006). EU-Funded Open Source Project gvSIG Launches v1.0. (Available online at <http://www.itbusinessedge.com/item/?ci=22753>, retrieved 09.10.2008).
- Lerner, J. and Tirole, J. (2002). Some simple economics of open source. *Journal of Industrial Economics*, 50(2):197–234.
- Löwe, P. (2008). Coming of Age: The Positive Legacy of FOSS GIS. In *the 2008 Geoinformatics Conference*.
- Marakas, G. M. (2006). *System Analysis and Design: an active approach*. McGraw-Hill/Irwin, New York, USA, 2. edition.
- Miller, C. C. (2006). A Beast in the Field: The Google Maps Mashup as GIS/2. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 41(3):187–199.
- Mitasova, H. and Neteler, M. (2004). GRASS as Open Source Free Software GIS: Accomplishments and Perspectives. *Transactions in GIS*, 8(2):1361–1682.

-
- Moreno-Sanchez, R., Anderson, G., Cruz, J., and Hayden, M. (2007). The potential for the use of open source software and open specifications in creating web-based cross-border health spatial information systems. *International Journal of Geographical Information Science*, 21(10):1135–1163.
- Neteler, M. and Raghavan, V. (2006). Advances in free software geographic information systems. *Journal of Informatics*, 3(2).
- Nichols, D. M. and Twidale, M. B. (2002). Usability and Open Source Software. (Available online at <http://opensource.mit.edu/papers/nicholstwidale1.pdf>).
- OGC (2000a). OpenGIS Geography Markup Language (GML) Encoding Standard 1.0. (Available online at <http://www.opengeospatial.org/standards/GML>).
- OGC (2000b). OpenGIS Web Map Server Interface Implementation Specification 1.0.0. (Available online at <http://www.opengeospatial.org/standards/wms>).
- OGC (2002a). OpenGIS Web Feature Service Implementation Specification 1.0.0. (Available online at <http://www.opengeospatial.org/standards/wfs>).
- OGC (2002b). Styled Layer Descriptor Implementation Specification. (Available online at <http://www.opengeospatial.org/standards/sld>).
- OpenBRR (2005). Business Readiness Rating for Open Source. (Available online at <http://www.openbrr.org/wiki/index.php/Downloads>).
- OSGeo (2008). GDAL/OGR Info Sheet. (Available online at http://www.osgeo.org/gdal_ogr).
- Peng, Z.-R. (1997). An Assessment of the Development of Internet GIS. In *the 1997 ESRI User Conference*.
- Peng, Z.-R. and Zhang, C. (2004). The roles of geography markup language (GML), scalable vector graphics (SVG), and Web feature service (WFS) specifications in the development of Internet geographic information systems (GIS). *Journal of Geographical Systems*, 6(2):95–116.
- Ramsey, P. (2007). A Survey of Open Source GIS. In *the 2007 FOSS4G Conference, Victoria, Canada* (available online at http://www.foss4g2007.org/presentations/view.php?abstract_id=136).
- Raymond, E. (1998). Goodbye, "free software"; hello, "open source". (Available online at <http://www.catb.org/~esr/open-source.html>).

- Ruffin, M. and Ebert, C. (2004). Using open source software in product development: A primer. *IEEE Software*, 21(1):82–86.
- Sánchez, W. (1999). Open Software in a Commercial Operating System. In *Proceedings of the 1999 USENIX Annual Technical Conference*, pages 139–142, Monterey, California, USA. USENIX Association.
- Scacchi, W., Feller, J., Fitzgerald, B., Hissam, S., and Lakhani, K. (2006). Understanding Free/Open Source Software Development Processes. *Software Process: Improvement and Practice*, 11(2):95–105.
- Shneiderman, B. and Plaisant, C. (2005). *Designing the User Interface*. Pearson Education, Inc., 4. edition.
- Souleyrette, R. R. and Anderson, M. D. (1998). Developing Small Area Network Planning Models Using Desktop GIS. *Journal of Urban Planning and Development*, 124(2):55–71.
- Stallman, R. (2007). The GNU GPL and the American Way. (Available online at <http://www.gnu.org/philosophy/gpl-american-way.html>).
- Steiniger, S. and Bocher, E. (2008). An Overview on Current Free and Open Source Desktop GIS Developments. Accepted for publication in *Int. J. of Geographical Information Science*, available online at http://www.geo.unizh.ch/publications/degen/sstein_foss_desktop_gis_overview.pdf.
- Strasser, T. C. (1995). Desktop GIS in libraries, technology and costs: A view from New York State. *The Journal of Academic Librarianship*, 21(4):278–282.
- Szczepanska, A. M., Bergquist, M., and Ljungberg, J. (2005). *Perspectives on Free and Open Source Software*, chapter 22, pages 431–446. The MIT Press, Cambridge, Massachusetts, 1. edition.
- The F.F.I.E. Council (2004). Risk Management of Free and Open Source Software. (Available online at http://www.ffiec.gov/ffiecinfobase/resources/info_sec/2006/frb-sr-04-17.pdf).
- Vivid Solutions (2008). JTS Topology Suite. (Available online at <http://www.vividsolutions.com/jts/jtshome.htm>).
- Wheeler, D. A. (2004). Why Open Source Software / Free Software (OSS/FS)? Look at the Numbers! (Available online at http://www.dwheeler.com/oss_fs_why.html).

- Wikipedia (2008). Retronym. (Available online at <http://en.wikipedia.org/wiki/Retronym>, retrieved 08.09.2008).
- Wikstrøm, M. and Tveite, H. (2005). PostgreSQL/PostGIS and MapServer compared to ArcSDE and ArcIMS in performance on large geographical data sets. *Kart og plan*, 3(2005):185–192.
- Ye, Y. and Kishida, K. (2003). Toward an Understanding of the Motivation of Open Source Software Developers. In *Proceedings of the 25th International Conference on Software Engineering*, volume 3, pages 419–429, Portland, Oregon, USA. IEEE Computer Society.
- Østerlie, T. and Jaccheri, L. (2007). A Critical Review of Software Engineering Research on Open Source Software Development. In *Proceedings of the 2nd AIS SIGSAND European Symposium on Systems Analysis and Design*, pages 12–20, Gdansk, Poland. Gdansk University Press.

APPENDIX

A Questions Used in the Interview

This list of questions was used during the interview-session. As the interview was conducted in Norwegian the list is not translated.

Dagens system (funksjonelle/tekniske)

- Hva slags programvare er dagens system basert på?
- Maskinvare systemet kjører på
- Operativsystem, i bruk nå og evt vurdert å bruke
- Dataformater i bruk (shapefiler, databaser (type), netttjenester)
- Funksjoner i bruk i dagens system
- Hvilke funksjoner er essensielle (dvs hvilke må være til stede for at det skal fungere)
- Hvilke funksjoner er ønskelige (dvs som finnes i dag, men som ikke er essensielle)
- Hvilke funksjoner som ikke finnes i dag er ønskelige? (dvs ønsker)
- Hvor mange brukere har systemet i dag?
- Hvordan er det satt opp (en installasjon pr maskin, tynnklient etc?)
- Hvor mye er systemet tilpasset? Og hvordan er det eventuelt gjort?
- Hvordan fungerer eventuelt eksisterende supportavtaler? Har man egen kompetanse på support?

Rettslige

- Ser man for seg å endre noe på programmet (endre kode, plug-ins etc)
Skal slike endringer kunne distribueres?
- Har man i dag noe "garanti" fra leverandør som dekker problemer grunnet feil i programmet?
Er det behov for dette?

Økonomiske

- Hva inkluderer dagens avtale (programvare, support, lisenser etc)
- Vurderer man å bruke penger på support til FOSS?
- Hva forventer man å spare? Sett i forhold til forventet kostnad ved innføring

Strategiske

Hvilke beskrivelser passer best for den tenkte bruken?

- Plattform for forretningskritiske prosesser?
- Langtidsinvolvering
- Kostnadsreduksjon
- Bruke for å utforske ny teknologi
- Grunnlag for videre utvikling og forretningsmodell
- Løse et midlertidig problem
- Gjøre seg uavhengig av proprietære løsninger og tilbydere
- Som et middel for å bedre sikkerhet
- Forskning- og utviklingsøyemed

Systemet i dag

- Kortfattet beskrivelse av systemet
- Antall brukere, geografisk spredning?
- Supportrutiner, hvordan gjøres dette
- Hvor viktig for bedriften? (i forhold til omsetning eller andre metrikker)
- Type data, rutiner for håndtering av disse
- Hvor lenge har systemet vært i bruk, hvor erfarne er brukerne med generelle konsepter i forhold til spesifikke metoder i dagens løsning?
- Hva er den gjennomgående holdningen til dagens system (fornøyd, greit nok, noe burde gjøres?)

B Evaluation Criteria, Tables

This appendix lists various tables with evaluation criteria, grouped by the type of requirements they are meant to measure.

Required functionality	Score		
	0	1	2
Required functionality present	Several Essential requirements missing	All Essential requirements fulfilled	All requirements fulfilled

Table B.1: *Criteria for evaluation of required functionality.*

Evolution direction	Score		
	0	1	2
Clear direction of product evolution	No thoughts found	Thoughts on evolution present, only partial match	Good match between thoughts and intended use

Table B.2: *Criteria for evaluation of evolution direction.*

Target platform	Score		
	0	1	2
MS Windows Supported	Not supported	Supported	All major OSes supported

Table B.3: *Criteria for evaluation of target platform compliance.*

Reliability	Score		
	0	1	2
Number of open bugs for the last 6 months	> 750	75– 750	< 75
Number of bugs fixed the last 6 months (compared to opened)	< 30%	30 – 70 %	> 70%
Age	< 3 months	3 months to 3 years	> 3 years
Stability	Numerous patches or releases	Stabilized production release, but old	Releases are mainly new functionality, but some fixes
History, known problems	Several problems known	No known major problem or crisis	History shows good management of crisis situations
Forkability, source of forking	Very likely to be forked in the future	Comes from a fork, low probability of forking	Very little chance of being forked. Does not come from a fork

Table B.4: *Criteria for evaluation of reliability, taken from OpenBRR (2005) and Atos Origin (2006).*

Maintainability	Score		
	0	1	2
Quality of source code	Not very readable code or poor quality	Readable, but not commented in detail	Readable and commented, using classic design patterns and an applied coding policy
Technological dispersion	Use of numerous different languages	One main language, modules in other languages for specific requirements	One unique language
Intrinsic complexity	Very complex code. Modifications require high level of expertise	Not very complex code, expertise still required	Simple code and design, easy to modify
Technical documentation	No documentation	Incomplete or old documentation	Detailed and up to date documentation

Table B.5: *Criteria for evaluation of maintainability, taken from Atos Origin (2006).*

Community	Score		
	0	1	2
Average volume of general mailing list last 6 months	< 50 messages/month	50–500 messages/month	> 500 messages/month
Number of unique code contributors last 6 months	< 7	7–30	> 30
Developers, identification, turnover	Less than 3 developers, not clearly identified	between 4 and 7 developers, or more unidentified developers with important turnover	More than 7 developers clearly identified, very stable team
Activity on bugs	Slow reactivity in forum/maillist or nothing regarding bug fixes in release notes	Detectable activity but without process clearly exposed, long reaction/resolution time	Strong reactivity based on roles and tasks assignment
Activity on functionality	No or little new functionality	Evolution driven by core team or user requests, no clearly defined process	Tool(s) to manage feature requests, strong interaction with roadmap
Activity on releases	Very weak activity on both production and development releases	Activity on production and development releases. Frequent minor releases (bug fixes)	Important activity with frequent minor releases and planned major releases relating to the roadmap

Table B.6: *Criteria for evaluation community activity, taken from OpenBRR (2005) and Atos Origin (2006).*

Support	Score		
	0	1	2
Average volume of general mailing list last 6 months	< 50 messages/month	50–500 messages/month	> 500 messages/month
Quality of professional support	No professional support	Installation support only	Installation + troubleshooting + integration/customization support
Documentation	No user documentation	Documentation exists, but shifted in time, is restricted to one language or is poorly detailed	Documentation always up to date, translated and possibly adapted to different target readers.
Support climate	Hostile, mainly RTFM*-replies	Answers given are general and only point to possible solutions. No follow-up	Contributors ask for logs, details and tries to investigate the particular case and solve the problem. Usually several replies and follow-ups

*Read The F**ing Manual

Table B.7: *Criteria for evaluation support availability, taken from OpenBRR (2005) and Atos Origin (2006).*

Longlivity	Score		
	0	1	2
Average volume of general mailing list last 6 months	< 50 messages/month	50–500 messages/month	> 500 messages/month
Sponsor	Software has no sponsor	Software has an unique sponsor, or is about to get one	Software is sponsored by industry

Table B.8: *Criteria for for evaluation of project longlivity, taken from OpenBRR (2005) and Atos Origin (2006).*

Flexible Maintenance	Score		
	0	1	2
Required functionality present	Several Essential requirements missing	All Essential requirements fulfilled	All requirements fulfilled
Project Driver	Individuals	Group or corporation	Independent foundation supported by corporations
Difficulty to enter core developer team	Anyone can enter	Rather difficult, must contribute accepted patches for some time	Only after being active outside committer for a while
Average time to fulfill open feature requests	> 6 months	3–6 months	< 3 months

Table B.9: *Criteria for for evaluation of wether the product offers flexible maintenance according to individual needs, taken from OpenBRR (2005).*

Cost reduction potential	Score		
	0	1	2
Required functionality present	Several Essential requirements missing	All Essential requirements fulfilled	All requirements fulfilled
Dependencies	Several dependencies to unknown software	Limited number of dependencies, to software considered stable	No dependencies, or dependencies controlled by developers
Support climate	Hostile, mainly RTFM*-replies	Answers given are general and only point to possible solutions. No follow-up	Contributors ask for logs, details and tries to investigate the particular case and solve the problem. Usually several replies and follow-ups

*Read The F**ing Manual

Table B.10: *Criteria for evaluation of cost reduction potential.*

Influence of development	Score		
	0	1	2
Clear direction of product evolution	Not found	Some thoughts found	Well documented
Sponsor	Software has no sponsor	Software has an unique sponsor, or is about to get one	Software is sponsored by industry
Dependencies	Several dependencies to unknown software	Limited number of dependencies, to software considered stable	No dependencies, or dependencies controlled by developers
Average time to fulfill open feature requests	> 6 months	3–6 months	< 3 months
Support climate	Hostile, mainly RTFM*-replies	Answers given are general and only point to possible solutions. No follow-up	Contributors ask for logs, details and tries to investigate the particular case and solve the problem. Usually several replies and follow-ups

*Read The F**ing Manual

Table B.11: *Criteria for for evaluation of how possible it is to influence the further development, some content taken from Atos Origin (2006).*

Decrease of proprietary dependencies	Score		
	0	1	2
Required functionality present	Several Essential requirements missing	All Essential requirements fulfilled	All requirements fulfilled
Clear direction of product evolution	Not found	Some thoughts found	Well documented
Sponsor	Software has no sponsor	Software has an unique sponsor, or is about to get one	Software is sponsored by industry

Table B.12: *Criteria for for evaluation of possibility to decrease proprietary dependencies, some content taken from Atos Origin (2006).*

C Unique Evaluation Criteria

These are the unique evaluation criteria used for the evaluation.

- Required functionality present
- Clear direction of product evolution
- MS Windows Supported
- Number of open bugs for the last 6 month
- Number of bugs fixed the last 6 months
- Age
- Stability
- History, known problems
- Forkability, source of forking
- Quality of source code
- Technological dispersion
- Intrinsic complexity
- Technical documentation
- Dependencies
- Average volume of mailing list last 6 months
- Number of code contributors last 6 months
- Developers, identification, turnover
- Activity on bugs
- Activity on functionality
- Average time to fulfill open feature requests
- Activity on releases
- Quality of professional support
- Documentation
- Support climate
- Sponsor
- Project Driver
- Difficulty to enter core developer team

D Identity Card and Evaluation Sheet Templates

Template for the Identity Card used in the evaluation.

General Information	
Name	...
Description	...
Source	...
Licenses	...
Website	...
Compatible OSes	...
Fork origin	...
Functional and Technical Aspects	
Technologies	...
Prerequisites	...
Detailed functionality	...
Roadmap	...
Bug tracking	...
Feature requests	...
Help and Support	
Documentation	...
Free support	...
Paid support	...
Synthesis	
General trend	...
Comments	...

Table D.1: *Identity Card template, adopted from the QSOS Framework (Atos Origin, 2006).*

Criterion	Score		
	0	1	2
Required functionality present			
Clear direction of product evolution			
MS Windows Supported			
Number of open bugs for the last 6 months			
Number of bugs fixed the last 6 months			
Age			
Stability			
History, known problems			
Forkability, source of forking			
Quality of source code			
Technological dispersion			
Intrinsic complexity			
Technical documentation			
Dependencies			
Average volume of mailing list last 6 months			
Number of code contributors last 6 months			
Developers, identification, turnover			
Activity on bugs			
Activity on functionality			
Average time to fulfill open feature requests			
Activity on releases			
Quality of professional support			
Documentation			
Support climate			
Sponsor			
Project Driver			
Difficulty to enter core developer team			

Table D.2: *Evaluation Sheet template.*

E Calculation of Requirement Scores

This appendix shows the tables used to calculate the scores for each of the 12 requirements based on the 27 unique criteria used.

	uDig	gvSIG	OpenJUMP
Required functionality			
Required functionality present	2.00	2.00	1.00
Average	2.00	2.00	1.00
Evolution direction			
Clear direction of product evolution	2.00	2.00	2.00
Average	2.00	2.00	2.00
Target platform			
MS Windows Supported	2.00	2.00	2.00
Average	2.00	2.00	2.00
Reliability			
Number of open bugs for the last 6 months	2.00	1.0	2.00
Number of bugs fixed the last 6 months (compared to opened)	0.00	2.0	1.00
Age	2.00	2.00	2.00
Stability	1.00	2.00	2.00
History, known problems	1.00	1.00	1.00
Forkability, source of forking	2.00	2.00	1.00
Average	1.33	1.67	1.50
Maintainability			
Quality of source code	2.00	2.00	2.00
Technological dispersion	1.00	1.00	2.00
Intrinsic complexity	1.00	1.00	2.00
Technical documentation	2.00	0.00	1.00
Average	1.50	1.00	1.75
Community			
Average volume of general mailing list last 6 months	0.00	1.00	1.00
Number of unique code contributors last 6 months	1.00	1.00	0.00
Developers, identification, turnover	2.00	1.00	2.00
Activity on bugs	1.00	1.00	1.00
Activity on functionality	2.00	1.00	2.00
Activity on releases	2.00	2.00	1.00
Average	1.33	1.17	1.17

Continued on next page..

Continued..

	uDig	gvSIG	OpenJUMP
Support			
Average volume of general mailing list last 6 months	0.00	1.00	1.00
Quality of professional support	2.00	2.00	2.00
Documentation	2.00	2.00	1.00
Support climate	2.00	2.00	2.00
Average	1.50	1.75	1.50
Longevity			
Average volume of general mailing list last 6 months	0.00	1.00	1.00
Sponsor	2.00	2.00	2.00
Average	1.00	1.50	1.50
Flexible Maintenance			
Required functionality present	2.00	2.00	1.00
Project Driver	2.00	1.00	2.00
Difficulty to enter core developer team	1.00	1.00	1.00
Average time to fulfill open feature requests	0.00	2.0	1.00
Average	1.25	1.50	1.25
Cost reduction potential			
Required functionality present	2.00	2.00	1.00
Dependencies	2.00	0.00	1.00
Support climate	2.00	2.00	2.00
Average	2.00	1.33	1.33
Influence of development			
Evolution of software	2.00	2.00	2.00
Sponsor	2.00	2.00	2.00
Dependencies	2.00	0.00	1.00
Support climate	2.00	2.00	2.00
Average time to fulfill open feature requests	0.00	2.00	1.00
Average	1.60	1.50	1.75
Decrease of proprietary dependencies			
Required functionality present	2.00	2.00	1.00
Evolution of software	2.00	2.00	2.00
Sponsor	2.00	2.00	2.00
Average	2.00	2.00	1.67

F FOS GIS History Links

Links to websites presenting the history of various FOS GIS projects.

GDAL <http://trac.osgeo.org/gdal/wiki/FAQGeneral#WhenGDALprojectwasstarted>

GRASS GIS <http://grass.itc.it/devel/grasshist.html>

KOSMO <http://en.wikipedia.org/wiki/Kosmo>

MapServer <http://en.wikipedia.org/wiki/Mapserver>

PostGIS <http://www.refractions.net/products/postgis/>

Quantum GIS http://www.foss4g2007.org/presentations/view.php?abstract_id=244

uDig <http://www.refractions.net/products/udig/>

G Links to Project Websites

Links to the project websites for the ten candidate FOS Desktop GIS projects evaluated are given here

GRASS – <http://grass.itc.it/>

Quantum GIS – <http://www.qgis.org/>

uDig – <http://udig.refractions.net/>

gvSIG – <http://www.gvsig.gva.es/>

SAGA – <http://www.saga-gis.org/en/index.html>

ILWIS – <http://www.itc.nl/ilwis/default.asp>

MapWindow GIS – <http://www.mapwindow.org/>

OpenJUMP – <http://openjump.org/wiki/show/HomePage>

KOSMO – <http://www.opengis.es/>

OrbisGIS – <http://orbisgis.cerma.archi.fr/>

H Calculation of Code Statistics

The webpage `hppt://www.ohloh.net` presents code statistics for a variety of FOSS projects, based on svn repository access. This ensures that statistics are up to date. However the code statistics includes "programming languages" such as HTML, XML and shell script. This means that the code percentages are a little distorted. In order to calculate the use of each actual programming language some modifications are made to the statistics presented on the *Code Analysis* page on ohloh.net.

The page lists a breakdown of Code Lines, Comment Lines, Comment Ratio, Blank Lines and Total Lines. The column Code Lines is the important one. The table is imported into a spreadsheet and the unnessecarry columns are deleted. Then "langauges" such as HTML and XML are removed, so that only actual programming langauges remain. Examples of removed languages are:

- HTML
- XML
- XML Schema
- TeX/LaTeX
- shell script
- CSS
- make
- DOS batch script
- MetaFont
- XSL Transformation
- SQL

The percentage of each of the remaining languages are then calculated. The tables on the next page shows the numbers used for this study.

uDig

Language	Code Lines	Percentage
Java	405681	99.24
Groovy	1036	0.25
Perl	772	0.19
JavaScript	599	0.15
Scala	528	0.13
Ruby	172	0.04
SUM	408788	100.00

Source: <http://www.ohloh.net/p/4677/analyses/latest>

gvSIG

Language	Code Lines	Percentage
Java	813519	76.48
C++	130710	12.29
C	51800	4.87
Python	49114	4.62
JavaScript	17771	1.67
Perl	670	0.06
Objective C	101	0.01
Haskell	56	0.01
D	2	0.00
SUM	1063743	100.00

Source: <http://www.ohloh.net/p/gvsig/analyses/latest>

OpenJUMP

Language	Code Lines	Percentage
Java	113621	100
SUM	113621	100

Source: <http://www.ohloh.net/p/9819/analyses/latest>

I Mailing List Statistics

This appendix lists the data used to calculate average mailing list volumes.

uDig

	User	Developer
June 2008	56	484
July 2008	29	260
August 2008	35	385
September 2008	41	438
October 2008	27	306
November 2008	44	247
AVERAGE	39	353

Sources: <http://lists.refractions.net/pipermail/udig-users/> and <http://lists.refractions.net/pipermail/udig-devel/>

gvSIG

	International	User (es)	Developer (es)
June 2008	145	297	81
July 2008	136	293	60
August 2008	101	197	42
September 2008	66	268	18
October 2008	239	432	132
November 2008	165	379	79
AVERAGE	142	311	69

Sources: Extracted manually from http://www.mail-archive.com/gvsig_usuarios@runas.cap.gva.es/maillist.html, http://www.mail-archive.com/gvsig_desarrolladores@runas.cap.gva.es/maillist.html and http://www.mail-archive.com/gvsig_internacional@runas.cap.gva.es/maillist.html

OpenJUMP

	User	Developer
June 2008	24	209
July 2008	21	184
August 2008	35	29
September 2008	85	54
October 2008	64	168
November 2008	123	138
AVERAGE	59	130

Sources: <http://groups.google.com/group/openjump-users/about> and http://sourceforge.net/mailarchive/forum.php?forum_name=jump-pilot-devel

