Evaluation Criteria for Free/Open Source Software Products

through Project Analysis

David Cruz¹, Thomas Wieland², Alexander Ziegler¹

¹ Technical University Munich
Institute for Computer Science
Boltzmannstr. 3, 85748 Garching, Germany
Phone: +49 (89) 289-17388, Fax: +49 (89) 289-17307
E-mail: {cruz,zieglera}@informatik.tu-muenchen.de

² University of Applied Sciences Coburg
Department of Electrical Engineering and Computer Science
96450 Coburg, Germany
E-mail: thomas.wieland@fh-coburg.de

Abstract. Today many companies decide to select Free/Open Source software (F/OSS) products for various purposes, for example because of economical or quality reasons. For many areas of application they can choose from a variety of packages provided by different communities. Introducing a software tool into a company – either for supporting a certain business process or for development of own products – may be more or less critical depending on the application purpose from being used in an explorative area up to supporting mission critical operations.

One of the most crucial problems in the usage of F/OSS is the actual decision to use a particular product. Although this is rather similar to the usual investment decisions, the consequences of an involvement in a software project are frequently underestimated. Whether a certain package is appropriate for the intended purposes in the given environment and is fitting to the over-all business goals of the company, is usually not determined systematically, but often arbitrarily. However many important hints resulting from the development context of an F/OSS product as well as other technical and economical considerations could be helpful to bring more transparency and determinism into the product decision process.

In this paper we give a systematic approach for supporting a decision to incorporate an F/OSS product into an enterprise. Systematic decision support is provided by a rational approach which facilitates evaluating and interpreting the relevant project circumstances of an F/OSS product to be chosen. By this means F/OSS projects can be critically reviewed in the light of the particular product requirements of the choosing organization. The article addresses software managers and engineers in industry and academia.
1 Project Circumstances Supporting a Product Decision

When a company or any other organization decides to select Free/Open Source software (in short: F/OSS) products, a selection process has to be undergone that is similar to that for commercial off the shelf-software, but differs in a number of aspects. Not only the functionality is now relevant, but also various conditions of the project that creates this product have to be taken into account. In this paper, we present a systematic approach for supporting a decision to utilize an F/OSS product by an enterprise. (Throughout this text we understand "Free/Open Source Software" as some piece of software that is released under a license approved by the Free Software Foundation or the Open Source Initiative.)

The project circumstances of the F/OSS project and its community have a clear influence on the product in its present and anticipated future form [SSR02], [SW03]. From the analysis of the project assumptions about its conditions, quality, and reliability can be made [CS02]. For example, information from the project source code repository can be used and interpreted to make several hypotheses concerning the state of the project [KS02]. Numerous essential characteristics of the software, like reliability, maintainability, or sustainability, cannot be identified by source code inspection alone, but have to include the environment in which it has been created – even if a fully detailed code review is possible within the given frame of resources. A thorough test of the software may reveal some other important facts, but is also just a technical snapshot. A company which is planning parts of its business to rely on an F/OSS package should thus conduct a systematic and detailed analysis before selecting a particular product. The central idea of our approach is provide some support for determining the requirements on an F/OSS product and evaluating the extent to which these requirements are actually met.

The evaluation and selection of an F/OSS package has some similarities with the conventional process of selecting commercial-of-the-shelf (COTS) software. As this has been described repeatedly and thoroughly (see, e.g., [Kon96], [MSB02]), we can take most of these results as known and emphasize the differences in particular. For not only the availability of the source code alone makes the F/OSS different to COTS software, but the entire creation environment requires a special approach and leads to some completely new implications.
2 Outline of the Presented Approach

Starting from possible scenarios for adoption of F/OSS in a company environment, the contribution deduces which possible requirements can be posed to the F/OSS product and its project environment in chapter 3. The requirements are grouped into several categories and are described in more detail in chapter 4. Furthermore chapter 4 also gives indicators on when the choosing company should take care that a specific requirement is met, if the scenarios for adoption are not enough in order to support such a decision. The paper then presents several investigable criteria that may be determined by analysing the F/OSS product and its surrounding project in chapter 5, also describing how the information for evaluating these criteria can be investigated. Finally, in chapter 6 we show how to use the criteria in order to determine whether a specific requirement posed at the F/OSS can be met.

The coherence of the above mentioned topics is visualised in Figure 1.

![Decision Support for F/OSS](image)

**Figure 1 Overview over decision support approach**

A user of the presented approach can utilize the use cases in order to classify his situation and determine the requirements that should be posed at the F/OSS. The chapter about the criteria helps the user in finding information that supports the systematic analysis of the project circumstances. Now the user is ready to interpret the collected information, determine if the necessary requirements have been met and decide whether to incorporate the F/OSS into company usage.

3 Scenario for Adoption of F/OSS

In the following plausible applications of F/OSS are presented. What scenarios for adoption of an F/OSS package are possible? Various applications of F/OSS in different domains are thinkable. Classical appli-
cation scenarios for F/OSS are web, mail or file server, firewalls, or embedded systems. Usual application models are simply using an F/OSS product for daily business, as design choice for developing a new software product, as a support providing business model etc. [GKL04], [HHMW04]. Furthermore, open source software products already serving in many purposes: for example in bioinformatics and genetic research [BWG04], in drug discovery [DeL05], or for medical image processing and visualisation [YA05]. In addition, there are strong arguments for using F/OSS in security critical domains (transparency of code, amount of testers and speed of bug fixing) [HPW02], [Pay02].

Several scenarios for adoption can motivate a manager to elect an F/OSS product for a company. The scenarios for adoption listed below can appear in various combinations. They are related with the requirements of section 4 in the following table.

<table>
<thead>
<tr>
<th>Scenario for Adoption</th>
<th>Related requirements of section 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use F/OSS as platform for a mission critical process.</td>
<td><strong>Functional</strong>: Functionality covered, clear evolution direction</td>
</tr>
<tr>
<td>(The F/OSS is existentially important for a company's</td>
<td><strong>Technical</strong>: Target platform supported, reliability, maintainability</td>
</tr>
<tr>
<td>core process, every problem causes considerable costs.)</td>
<td><strong>Organisational</strong>: Maintenance active, sufficient support, long life existence</td>
</tr>
<tr>
<td></td>
<td><strong>Economical</strong>: Sustainability, flexible maintainability according to individual needs</td>
</tr>
<tr>
<td></td>
<td><strong>Political</strong>: Possibility for influencing further development according to individual needs, decrease of proprietary dependencies, transparency over security</td>
</tr>
<tr>
<td>Use F/OSS with a long term consideration.</td>
<td><strong>Functional</strong>: Clear direction of product evolution</td>
</tr>
<tr>
<td>(The F/OSS is part of the technological strategy of an</td>
<td><strong>Technical</strong>: Target platforms supported, maintainability</td>
</tr>
<tr>
<td>enterprise.)</td>
<td><strong>Organisational</strong>: Community exists, long life existence</td>
</tr>
<tr>
<td></td>
<td><strong>Economical</strong>: Sustainability, protection of investment</td>
</tr>
<tr>
<td></td>
<td><strong>Political</strong>: Possibility for influencing further development with respect to individual needs</td>
</tr>
<tr>
<td>Use F/OSS as cost reduction model.</td>
<td><strong>Functional</strong>: Required functionality covered, clear direction of product evolution</td>
</tr>
<tr>
<td>(The F/OSS replaces a pro-</td>
<td><strong>Economical</strong>: Sustainability, protection of investment, cost reduction,</td>
</tr>
<tr>
<td>Scenario for Adoption</td>
<td>Related requirements of section 4</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>proprietary product and the license fee is saved.</td>
<td>division of development costs</td>
</tr>
<tr>
<td><strong>Political:</strong> Possibility for influencing further development with respect to individual needs</td>
<td></td>
</tr>
<tr>
<td><strong>Use F/OSS as exploration object, for example for technology.</strong></td>
<td><strong>Organisational:</strong> community exists</td>
</tr>
<tr>
<td>(The F/OSS is used as a study object to learn about new technologies or paradigms.)</td>
<td><strong>Economical:</strong> Increasing know-how</td>
</tr>
<tr>
<td><strong>Use F/OSS as exhibition prototype.</strong></td>
<td><strong>Functional:</strong> Required functionality covered</td>
</tr>
<tr>
<td>(The F/OSS supports an exhibition of a product owned by the company or is exhibited as its own further development.)</td>
<td><strong>Organisational:</strong> Community exists</td>
</tr>
<tr>
<td><strong>Economical:</strong> Flexible maintainability according to individual needs, quick availability</td>
<td></td>
</tr>
<tr>
<td><strong>Political:</strong> Publicity, marketing effects</td>
<td></td>
</tr>
<tr>
<td><strong>Use F/OSS as base line for further development and business model.</strong></td>
<td><strong>Technical:</strong> Target platforms supported, Maintainability</td>
</tr>
<tr>
<td>(The F/OSS is the base line for further proprietary development to be sold or to be subject of support contracts.)</td>
<td><strong>Legal:</strong> No copyleft, liability, patent infringements, reselling obligations</td>
</tr>
<tr>
<td><strong>Economical:</strong> Flexible maintenance according to individual needs, increasing know-how</td>
<td></td>
</tr>
<tr>
<td><strong>Political:</strong> Transparency over security</td>
<td></td>
</tr>
<tr>
<td><strong>Use F/OSS to bridge a temporary bottleneck.</strong></td>
<td><strong>Functional:</strong> Required functionality covered</td>
</tr>
<tr>
<td>(The F/OSS fills a temporary gap in the migration from one software package to another.)</td>
<td><strong>Technical:</strong> Reliability</td>
</tr>
<tr>
<td><strong>Organisational:</strong> Sufficient support available</td>
<td></td>
</tr>
<tr>
<td><strong>Economical:</strong> Flexible maintainability according to individual needs, quick availability</td>
<td></td>
</tr>
<tr>
<td>Scenario for Adoption</td>
<td>Related requirements of section 4</td>
</tr>
<tr>
<td>----------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Political:</strong> Decrease of proprietary dependencies</td>
<td><strong>Functional:</strong> Required functionality covered, clear direction of product</td>
</tr>
<tr>
<td><strong>Use F/OSS for becoming independent of proprietary solutions and providers.</strong></td>
<td><strong>Technical:</strong> Reliability</td>
</tr>
<tr>
<td>(The F/OSS helps to resolve dependencies from software product companies with big market power and low customer orientation.)</td>
<td><strong>Organisational:</strong> Community exists, maintenance active, sufficient support available</td>
</tr>
<tr>
<td><strong>Economical:</strong> Sustainability, flexible maintenance according to individual needs</td>
<td><strong>Political:</strong> Possibility for influencing further development with respect to individual needs, decrease of proprietary dependencies</td>
</tr>
<tr>
<td><strong>Use F/OSS to gain transparency concerning safety and security.</strong></td>
<td><strong>Functional:</strong> required functionality covered</td>
</tr>
<tr>
<td>(The F/OSS is used to become independent from software companies not trustworthy in security concerns.)</td>
<td><strong>Economical:</strong> Sustainability</td>
</tr>
<tr>
<td><strong>Political:</strong> Decrease of proprietary dependencies, transparency over security</td>
<td><strong>Political:</strong> Decrease of proprietary dependencies</td>
</tr>
<tr>
<td><strong>Use F/OSS for research purposes.</strong></td>
<td><strong>Functional:</strong> Required functionality covered, clear direction of product</td>
</tr>
<tr>
<td>(The F/OSS is used to support research activities.)</td>
<td><strong>Technical:</strong> Reliability</td>
</tr>
<tr>
<td><strong>Organisational:</strong> Sufficient support available</td>
<td><strong>Economical:</strong> Flexible maintenance according to individual needs</td>
</tr>
<tr>
<td><strong>Political:</strong> Possibility for influencing further development with respect to individual needs</td>
<td><strong>Political:</strong> Possibility for influencing further development with respect to individual needs</td>
</tr>
<tr>
<td><strong>Use F/OSS as a CASE tool.</strong> (The F/OSS is used for developing software.)</td>
<td><strong>Functional:</strong> Required functionality covered, clear direction of product</td>
</tr>
<tr>
<td><strong>Technical:</strong> Target platforms supported, reliability</td>
<td><strong>Organisational:</strong> Community exists, maintenance active, sufficient support available</td>
</tr>
</tbody>
</table>
### Scenario for Adoption Related requirements of section 4

<table>
<thead>
<tr>
<th>Support available</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Economical:</strong> Sustainability, protection of investment, increase productivity, flexible maintainability according to individual needs, cost reduction, division of development costs</td>
</tr>
<tr>
<td><strong>Political:</strong> Possibility for influencing further development with respect to individual needs, decrease of proprietary dependencies</td>
</tr>
</tbody>
</table>

All these scenarios for adoption of F/OSS will occur in various combinations and are multiply related to target-oriented requirements for an F/OSS product and project like shown in the table above. These requirements will be detailed in the following chapter.

## 4 Requirements on Free/Open Source Software

This section discusses the question which requirements on free/open source software are reasonable. The close investigation of an F/OSS project can reveal many interesting details that clearly influence the decision in favour or against this software. But most of these must be viewed in the light of what the selecting company actually needs. The requirements give the chance to establish weights for the criteria laying particular emphasis on one aspect while more or less ignoring another one. In the same way the project-specific data are classified in several categories – which we will describe in full detail in the next section – also the company-specific requirements range from very technical and functional aspects to economical and even political characteristics. It is important that these requirements are elaborated in sufficient details prior to the selection procedure. Only if the features of the entire system, in which the F/OSS shall be embedded, are clear, an unbiased selection can be done. It can, however, be recommended to keep at least the functional requirements as simple and broad as possible. The required features should be listed by priorities. In this list, the number of unconditional features should be kept relatively short. Very strong functional requirements may limit the range unnecessarily.

In this section we want to present a concise overview over various types of requirements that may affect the software. Some of them must be described item by item, like the functional requirements. Others are only qualitative aspects that should be arranged on a scale from "negligible" to "essential". For each re-
quirement category, we give a brief description and explain, whether a particular evaluation process
should lay emphasis on it. So “indication” means the circumstances under which this requirement should
be added to the evaluator's catalogue and traced furtheron.

4.1 Functional Requirements

The most important requirement is certainly the functionality, the features the software has to cover. A
schematic list for this aspect cannot be given, for it depends ultimately on the purpose for which the par-
ticular program or library is intended.

Required Functionality Covered

Description: The software comprises all the features it is supposed to. The coverage is considered suffi-
cient if at least all the requirements indicated as "essential" are fulfilled and the respective
features are actually working.

Indication: This requirement is always crucial for it is the functionality why a particular piece of soft-
ware is actually selected and used.

Clear Direction of Product Evolution Recognizable

Description: The community has developed clear thoughts and plans which features will be changed or
added in the future. Such a plan is sometimes also called a "roadmap". So it is obvious
which additional functionality may be expected in which version. A time frame is desirable
but not mandatory.

Indication: When deciding for a particular product, one has to keep in mind that this may be not a deci-
sion just for the moment (unless it is only used for demonstration purposes or short-term
experiments). Commonly the selected software is integrated with own software components
and systems such that a strong mutual dependence is generated. So the functional require-
ments should not only comprise the current ones, but also the ones of future version that
can be planned or anticipated today. This requirement makes sense only if there is already a
clear view of the future usage of the software. In this case, it can increase the likelihood that
the selection of today is still right tomorrow.
4.2 Technical Requirements

Besides the mere functional requirements there are often numerous non-functional ones that are yet no
less technical. When designing a piece of software, a particular technical environment in which it is sup-
posed to run later is always expected. This environment (or maybe there are several of them) poses cer-
tain technical constraints on components, libraries, etc. the software depends on.

Target Platforms Supported
Description: The software supports all the platforms on which it is intended to be used. "Platform" in
this context can mean the operating system and required libraries, but also virtual environ-
ments like the Java platform.

Indication: This requirement is essential for most usage scenarios. If a package does not run on the
target platform, it is simply worthless for this purpose. The notion of the platform should,
however, be defined carefully to cover all mutual dependencies. Platforms that are not in
use today, but anticipated for the future, should also be taken into account.

Reliability
Description: The software is mature enough that all required features are working satisfactorily and
robust with respect to expected and unexpected failures.

Indication: Reliability is clearly mostly an issue. The exact degree of reliability depends on the usage
context, i.e. how reliable the entire software system is supposed to be. First this has to be
defined, before it can be decided if the F/OSS under question is reliable enough or not.

Maintainability
Description: This aspect means the ease with which a software system can be modified to correct faults,
 improve performance, or other attributes, or adapt to a changed environment [IEEE90].

Indication: The importance of this requirement depends on the purpose for which the software is used.
Experimental or short-term projects may accept that they only take one snapshot of the
package and make no changes after the end of the project. Productive usage certainly re-
quires a much larger degree of maintainability.
4.3 Organizational Requirements

Requirements which a company may expect from the community that develops or maintains the software of interest are grouped within this document as organizational requirements. The simplest example for such a requirement is that such a community exists at all, for there are many F/OSS systems on which the development has stopped and which aren't maintained any more. This section identifies such requirements, describes their meaning within this document and points at simple indications which can show a company the relevance of such a requirement.

Community exists

Description: As stated above the existence of a community is the simplest organizational requirement.

Within this document it means that there are persons which can be contacted to exchange information about the software. These persons must not necessarily be developers. For the community to exist, it is also sufficient if there are users with whom information about the software may be exchanged.

Indication: The only scenarios when this requirement may be unnecessary for the company are when the company developers have a full understanding of the software or when the software is only to be used for non-critical purposes. Another option is that the software product is so unique that the company intends to continue the project under its own auspices.

Product evolves

Description: The software is being adapted to the changing environment in which it is used as well as to changing requirements.

Indication: If the company is not able to maintain the software itself completely and it is willing to use the software over a longer period of time, then this requirement should be met.

Sufficient support available

Description: When technical difficulties of any kind arise in working with (using or developing) the software there is a possibility of contacting someone who can help within an acceptable period of time.

Indication: Indications for this requirement could be, if the usage of the software is nontrivial or requires special knowledge. This requirement is also important, if the company wants to further develop the software and the software is complex either in size or in difficulty.
Long life existence

Description: The software and the developing/maintaining community are bound to still exist in a long period of time.

Indication: When the company does not have the means of maintaining and further developing the software itself and the software will be used in a company critical area for a long period of time, then this requirement should be met.

Compatible development process

Description: The development process of the F/OSS and the development process of the company can be integrated to work as a whole. Sometimes it is not possible to combine the development processes, for example, when there are fixed deadlines for releases within the company process and the release decision for the F/OSS is made by a member of the community [LUS04].

Indication: In some use cases the company needs to take part in the development of the F/OSS. If the interaction with the development process of the free/open source community is high, i.e. the development contribution of the company is nontrivial, then this requirement must be taken into consideration.

4.4 Legal Requirements

The possibility of having legal requirements may seem strange at first, but given the very different legal implications given by the different F/OSS licences it is essential that a company is certain about its own expectations. This section will address some of the legal requirements to keep in mind when deciding on whether to use F/OSS.

No copyleft effect for add-ons or combinations

Description: The licence does not oblige the company to licence own add-ons under the same licence as the F/OSS. There are no obligations to licence combinations of the F/OSS and proprietary software under the same licence as the F/OSS.

Indication: If the company wants to develop add-ons for the F/OSS, which should not have to be free/open source then it is important for the F/OSS licence to allow such add-ons. The company may have an interest in keeping the licence of add-ons “closed source” if it wants
to sell the add-ons or if company confidential information is inside the source code. The same reasoning applies to combinations of proprietary software with the F/OSS.

**No liability for third party code**

*Description:* In many countries the law will require that the company selling a product will be liable for its correct functionality. Given that the company selling the product did not write parts of the code, it may not want to be liable for the foreign code parts.

*Indication:* Depending on the country in which the product is sold, this is a very strict requirement which may not be fulfilled. Therefore this requirement should be investigated wisely. Indications are areas where the integrity of humans or animals is endangered (e.g. transportation) or areas in which large amounts of money are involved in the software (e.g. financial software).

**No patent infringements**

*Description:* This requirement demands that the use of the F/OSS in question does not collide with any patent restrictions.

*Indication:* When the F/OSS is to be used for mission critical purposes, this requirement is absolutely necessary. For other uses the need for this requirement has to be determined for each individual case.

### 4.5 Economical Requirements

Economical requirements are relevant, because the usage of an F/OSS is embedded in a context of an organisation under financial constraints which cannot be considered less important than technical issues because they are existential for the F/OSS using company. Therefore topics like sustainability, protection of investment, increase of productivity, flexible maintenance according to individual needs, quick availability, increasing know-how, cost reduction and division of development costs have to be discussed in this contribution. Different from the above stated requirements this is carried out from the particular point of view of the target environment.

**Sustainability of the usage of the F/OSS**

*Description:* Sustainability means long term availability of product, support and maintenance. For example a firewall for an IT company should be sustainable.
Indication: It is important that the maintenance of the F/OSS product is granted for a long time if the purpose is to use it for several years. This can be achieved either by the using company itself, by a third party support company or by the open source community under consideration.

Protection of investment for migrating to the F/OSS product

Description: The financial and time effort for migrating to the free/open source software product has to be prepaid. The purpose is to regain the invested money through the usage of the F/OSS product. For example the effort for evaluating, testing, deploying, and training an F/OSS should amortize through its usage.

Indication: The costs for the migration to the F/OSS product may be significant and must be amortised to get that business case profitable and acceptable.

Increase productivity by usage of the F/OSS

Description: Increasing productivity means improving the relation between business values and working time. For example using an e-mail client should support its users in their easy communication with each other and save time for further useful activities in the company.

Indication: The processes supported by the F/OSS should increase productivity to justify the afforded investment for migrating to the open source product.

Flexible maintainability of the F/OSS according to individual needs

Description: The F/OSS is easily customizable to fulfil functional and non-functional requirements of the company.

Indication: Flexible maintainability according to individual needs is often necessary to optimize the efficiency of the usage of the open source product

Quick availability of the F/OSS

Description: The F/OSS is ready for use easily and in a short time. For example an email client should be easily downloadable for evaluation purposes.

Indication: If the possibility of use stands under time pressure, quick availability is most important

Increasing know-how for the internal staff by studying the F/OSS
Description: The F/OSS provides inherent technological progress to be studied by the internal staff. For example a file server with a sophisticated technology or implementing a new paradigm can be useful for the internal staff to study and gain knowledge about these topics.

Indication: The provision of additional know-how may be important for a company exploring new possibilities to enrich products with state of the art technology or to grow into a new technological segment. Increasing know-how of the staff means improving an important competition factor.

**Cost reduction through usage of the F/OSS product**

Description: The effort for running the free/open source software product in daily business is lower than the effort for the former solution. For example running an F/OS anti virus software product saves the license fee with respect to a proprietary one.

Indication: Maybe a decrease of running costs is important for a company to stay competitive.

**Division of development costs through usage of the F/OSS product**

Description: Sometimes a company cannot afford the money and time for a needed software solution alone and searches for others with similar interests and purposes. For example smaller companies depend on F/OSS to share development costs with others when the market is too small for a proprietary product and individual development is too expensive.

Indication: A solution is needed, where a trade off between lower costs and suboptimal customizing with respect to individual needs due to other project members is accepted.

4.6 Political Requirements

Like economical topics, political issues have to be considered very carefully because neither the F/OSS and its community nor the choosing company can be considered independent from the rest of the IT world the rapid movements and changes of which always have side effects for related companies. For this reason factors like possibility for influencing further development with respect to individual needs, decrease of proprietary dependencies, transparency over safety and security and publicity and marketing effects have to be looked at.

**Possibility for influencing further development of the F/OSS with respect to individual needs**
Description: An F/OSS package often has to be customized according to company specific requirements. Therefore a sufficient willingness in the corresponding open source community is necessary. For example if a company needs individual features for its F/OSS file server and has no own development resources, there should be a satisfactory possibility to place those interest in the community.

Indication: It must be recognizable that there are adequate possibilities to place the companies feature interests in the community maintaining the open source product under consideration.

Decrease of proprietary dependencies through usage of the F/OSS

Description: Often organisations using proprietary software products depend strongly on decisions made by the product provider and their needs aren’t respected sufficiently. For example if a large product vendor refuses to implement features of an Email client which are necessary a smaller part of its customer community, choosing an adequate F/OSS could be a solution.

Indication: Independency from large software companies and their market power shall be achieved through the use of the F/OSS by being able to influence the maintenance of the product for the user company itself.

Transparency over security through usage of F/OSS

Description: For information security reasons, transparency about the source code is needed for a software product that deals with a company’s information assets. For example, if it is impossible to evaluate a proprietary operating system concerning security leaks, choosing an F/OSS platform could be an option because of source code availability and transparency.

Indication: To be sure about information safety and security it is needed to have insight to the source code of the according software product, for example by the internal staff.

Publicity, marketing effects through usage of and participating in F/OSS

Description: It is publicly known that a company or organisation participates in an open source project. For example if a large software vendor which doesn’t provide an operating system participates publicly from a reputable F/OSS operating system there is a publicity effect.
Indication: Sometimes it may be important for the reputation of a company to be known as a participant in and user of a well-known open source project and product.

After the above stated possible requirements for F/OSS it is necessary to present criteria which give hints about their fulfilment. These criteria follow in the section below. The connection between the criteria and the requirements is created in section 6.

5 Investigable Data about an Free/Open Source Software Product

There is a large number of aspects of a particular F/OSS project that can affect the decision in favour or against using its software. Some of them are more abstract and hard to determine, while others are easy to be evaluated quantitatively. In this section we discuss all issues about a software project that are relevant to our approach and thus regarded as investigable. With the help of these data, the fulfilment of the respective requirements can be checked systematically. For each criterion, we not only give a brief description, but also point out how the fulfilment of this criterion could be determined or measured, respectively.

5.1 Functional Criteria

The functional criteria seem to be determined most simply. Almost every project has a description page that lists the features of the software. Sometimes, however, the terminology must first be learned to comprehend the feature descriptions. Some other time the feature list is far from being complete such that the actual features must be explored by running demos, test suites, or by analysing reference installations. The functionality is always a crucial aspect when selecting a certain software product. In case of F/OSS, however, there is no general way to determine it exhaustively. It always depends on the actual project and the type of software how the feature list is set up.

5.2 Technical Criteria

There are numerous technical aspects that can serve as selection criteria. The most relevant ones are discussed in the following:
Actual number of bugs
Description: The number of bugs (failures, errors, inconsistencies) which are inherent in the F/OSS under consideration.
Measurement: This number can only be determined if the project maintains a public bug list. Common project portals (e.g. Sourceforge, www.sf.net) offer and support this feature, but no all projects make use of it. Larger projects that rely on a bug tracking system (e.g. BugZilla, www.bugzilla.org) usually publish these data in full detail.
But even if there is a public bug list, it is not guaranteed that this reflects really the actual number of bugs. Often bugs that are noticed by a developer, are not reported formally, but fixed immediately. Afterwards only a message in the mailing list or a comment in the check-in protocol reminds of this program error.
Related issues are the number of fixed bugs that can be determined similarly and the number of reported bugs. The latter means program errors that are noticed and reported by others than the developers themselves.

Number of open feature requests
Description: The number of request for new or modified features that have not been implemented.
Measurement: Like the number of bugs, the number of open feature request can best be determined from public bug tracking systems, either via a portal site or a dedicated installation. Usually for each request its status is denoted here, e.g. open, assigned, realized, or discarded.

Code metrics
Description: Statistical data about the source code, e.g. number of files, number of classes, average length of functions, fraction of comment lines vs. code lines.
Measurement: This data can be evaluated automatically by shell scripts and other specialized tools.

Frequency of changes
Description: Information about how often each project file has been changed and how frequent code is checked in to the repository.
Measurement: Usually the code repository is publicly accessible. So it can be evaluated by appropriate scripts from the repository log data.

Dependencies on other software
5.3 Organizational Criteria

The investigable information about the organization, i.e. the organization of the community developing the F/OSS in question, is the following:

Number of developers

Description: The number of developers which are involved in the development of the F/OSS in question.

Measurement: In most Open Source Portals the project page will show the amount of developers involved in the project.

Number of testers

Description: This document defines a tester as any person giving feedback, i.e. reporting bugs to the project developers.

Measurement: Usually every F/OSS project has a means of letting users submit bug reports to the project developers. Bugs may be reported using a special bug reporting tool as BugZilla (www.bugzilla.org) or simply through a forum or mailing list. The amount of testers is determined by exploring the means used to report bugs and counting the number of different identities reporting bugs, not counting the project developers.

Number of users

Description: The amount of persons or companies using the software.

Measurement: Unfortunately this information can only be estimated. An estimate of this information may be obtained by observing the number of downloads of the software. Some Open Source Portals provide statistical information for the project in which the number of downloads are included. If the Portal or Site does not provide this information, it may be possible to ask the project leader. This would also give an insight on the reaction to user
questions. Another way of obtaining an estimate of the amount of users is to take a look at
the user mailing lists or user discussion forums, if available.

Development process patterns

Description: All forms of development obey to a set of rules. The use of certain rules suggests that the
development process follows a special development pattern.

Measurement: The rules of the development in an F/OSS project are usually described on the project site.
If a developer wants to contribute to an F/OSS project, it is common to send a patch to a
committer or the mailing list. The patch will then be reviewed and only checked into the
repository if its quality satisfies the reviewer(s). Further important rules may also be part
of the development process. If the rules are not published on the project site, the simplest
solution is to ask the project manager.

Skills of the developer community

Description: The know-how and experience of the developers.

Measurement: By following discussions of the developers in mailing lists, an expert of the area can de-
termine, whether the developers have the expertise in the area of the use of the developed
software (e.g. by determining, whether the correct terms are being used).

5.4 Legal Criteria

If the F/OSS product in question has a well known licence, some of the legal properties of an F/OSS
product can rapidly be determined by consulting websites and publications. For uncommon licences it is
usually wise to seek legal advice. Some legal criteria, which can eventually be rapidly determined is listed
below:

Copyleft effect of licence

Description: Whether the licence of the F/OSS implies that combinations of the F/OSS and other SW
must be licensed under the licence of the F/OSS licence.

Measurement: The licence of the F/OSS is usually presented on the project site and it is usually packaged
with the software when downloaded. As already mentioned, some properties (such as the
copyleft effect of a licence) of well known licences have been thoroughly discussed and
are thus publicly available. As an example the copyleft effect of the GNU GPL is publicly known.

**Liability when reselling**

**Description:** Weather the company selling a product containing the F/OSS or parts of it is liable for errors in the F/OSS and to what degree.

**Measurement:** Determining the liability the company has to adopt for the F/OSS is done in three steps. Most F/OSS licences disclaim any liability for the software. The first step is to determine if this is the case for the F/OSS in question. Next it should be found out if the county specific laws allow all liability to be disclaimed. As a third step it is useful to find out if the company itself is allowed to disclaim the liability for the third party software by the country specific laws.

### 5.5 Economical Criteria

**Is there sufficient availability of development resources?**

**Description:** What amount of infrastructure and resources through participating companies can be estimated? How much do large companies support the open source product with human resources, technical infrastructure and know-how? The better the support through resources, the better the vitality of the F/OSS.

**Measurement:** Often but not mandatory the project home page contains hints which organisations or companies participate in the project.

**What are the estimated migration efforts?**

**Description:** When migrating to an open source product sometimes a business should be calculated. The effort of time and money has to be estimated and compared with the expected benefit of the usage of the open source product. The fewer the efforts for the migration, the lower the financial risks.

**Measurement:** A cost benefit analysis considering and estimating the man power, Know-how, time, money and effort needed to migrate to, train the staff for and roll out the F/OSS and the expected benefit in form of for example cost reduction or increasing productivity should
What are the estimated monthly costs?

Description: How much are the running cost and effort for using the open source product for example in form of maintenance or in house support? The fewer the running costs, the better the long term economic efficiency.

Measurement: An analysis considering the running efforts and their costs for in house maintenance and support of the F/OSS has to be made.

5.6 Political Criteria

Which well reputed companies are involved?

Description: Which well known companies support the open source project with man power, technical resources, Know-how and political influence? The broader the political support for the F/OSS, the less risky it is to rely on it with a long term consideration.

Measurement: On the project home page and in the relevant IT press can be learned about the overall support for and interests in the F/OSS by several well known parties of the community.

Which dependencies on other F/OSS exist?

Description: Are there any dependencies from other open source products for running the F/OSS under consideration and what effort can be estimated for respecting them? The more the dependencies, the less calculable is the risk to rely on it.

Measurement: On the project home page should be links leading to existing dependent products.

In how many languages documentation exists?

Description: Is there any user documentation for the open source product and in which and how many languages is it available? The better and multi lingual the documentation, the broader is the usage and the larger the community. This again is a liveliness criterion.

Measurement: On the project home page should be links leading to existing documentation.

How is the climate in discussion forums?

Description: Is there a constructive progress oriented discussion climate in the forums? Are there helpful advices dominating or “rtfm”-like destructive answers? The more constructive
the discussion climate, the more progressive the overall attitude the more liveliness of
the F/OSS can be deduced.

Measurement: Look at the project discussion forums and consider the discussion threads.

Are there any large contributing companies about to be merged, bought in or to be insolvent?
Description: Are there any significant external changes in political, environmental or any other im-
portant circumstances to be expected for the open source project and product? The more
large contributors there are about to be retired from the project, the less is the liveliness
to be estimated and the more risky it will be to rely on it with a long term consideration.
Measurement: On the project home page and in the relevant IT press can be learned about the overall
support for and interests in the F/OSS by several well known parties of the community
as well as movements and purposes at the finance market.

Are there any large contributing companies about to have a strategic or executive board change?
Description: Are there any significant internal changes in political, environmental or any other impor-
tant circumstances to be expected for the open source project and product? The more
large contributors there are about to be retired from the project, the less is the liveliness
to be estimated and the more risky it will be to rely on it with a long term consideration.
Measurement: On the project home page and in the relevant IT press can be learned about the overall
support for and interests in the F/OSS by several well known parties of the community
and their enterprise strategic movements and purposes.

6 Fulfilment of Requirements

After having discussed possible requirements for an F/OSS in chapter 4, several investigable criteria for
evaluating an F/OSS have been presented in chapter 5, the main question is now how with the help of this
information the fulfilment of requirements can be estimated? Starting from the possible requirements it is
shown how an interpretation of the according F/OSS criteria can be carried out to support a product deci-
sion. The following creates the connection between the requirements described in section 3 and the criteria
discussed in section 5. For each requirement, we check which criteria are suitable to confirm its ful-
filment and give hints how these criteria could be applied or derived.
6.1 Fulfilment of Functional Requirements

The description page of a project usually contains a list of the features the software supports so they can be compared with the required functionality. But this relationship between features and functionality is neither necessary nor is it sufficient. On the one hand, there may be a couple of features which are listed, but which do not work properly (yet). On the other hand, some required qualities may be missing on list, although they are actually implemented and working. Both kinds of errors are hard to overcome. The first one is usually more serious for the selection of a program that does not work as it is supposed to may cause severe problems.

The detection and clearance of these errors depends on the amount of time and resources the company is willing and able to invest in the selection procedure of each package. A step-wise approach can be recommended:

1. Compare the list of required features and list of reportedly available features. If essential features are missing and further test resources are not assigned, the product has to be discarded.
2. Check if the reportedly available features are really implemented and are working properly. This can be done by defining and realizing appropriate test cases and by looking into the source code.
3. If features are missing, but test resources are on hand, check if these features are really absent. Again, this can be done by respective test cases or code inspection.
4. If required features are confirmed to be missing, estimate the effort and expenses for adding an own implementation.

6.2 Fulfilment of Technical Requirements

Actual number of bugs

It is difficult to interpret the number of bugs objectively. A larger number can mean are more buggy or instable product. But it can also mean that there are a lot of users that find many bugs by running the software on various platforms. Similarly, a small number can mean that there are simply no users that could find any bugs. So before drawing any conclusions about a certain project with respect to its number of bugs, the quality of these bugs should be analysed. If the project maintains no formal bug tracking, the mailing list commonly gives good hints on this issue.

Number of open feature requests
In turn, the number of open feature requests gives a good indication how vivid the user community is and how satisfied the users are with the software. A larger number usually means a better user echo.

**Code metrics**

Code metric data can give an impression about the complexity of a certain project. If the relationship of code lines vs. comment lines is, e.g., rather bad, there may be problem with maintainability or reliability. If the functions have more than 100 lines in average, the code quality is normally not the best such that stability and reliability may be affected.

**Dependencies on other software**

If the software product depends on several other projects which in turn depend on further packages themselves, the entire configuration complexity is usually quite high. It is not recommended to use such a package for mission critical purposes.

### 6.3 Fulfilment of Organizational Requirements

Drawing conclusions about the organization of the F/OSS project always has to be done by instinct. But above criteria can help in estimating whether and to what degree some of the requirements stated above can be met.

**Community exists**

In order to determine the existence of the community one of the following criteria has to be fulfilled: The number of developers or the number of users or the number of testers has to be greater than zero.

**Product evolves**

For the F/OSS product to evolve, its source code has to change. Therefore the existence of a community or even discussions in developer mailing lists are not enough to show that the product really evolves. To find out if the code really changes, the only possibility is to monitor the frequency of changes to the code repository. If a more detailed analysis is within budget and time constraints, the monitoring can be broadened to verify new development, reengineering, bug-fixing, testing, and other special cases.

**Sufficient support available**

For sufficient support to exist, as a first requirement a community has to exist. Other criteria aiding with the fulfilment of this requirement are the existence of documentation and the climate in discussion forums or mailing lists.
Long life existence

If some F/OSS has a large community, it insinuates that there is great interest in using such software. Thus, the users of the software are bound to accept investments in order to be able to continue using it. Another criterion which signalizes a long life existence is the involvement of well reputed companies, because these companies are also bound to protect their investments made in the software.

Compatible development process

The development process pattern of the F/OSS indicates whether it can be combined with the company development process. It is important to also analyse the development processes of other software, the F/OSS in question depends on. Usually this software is also open source.

6.4 Fulfilment of Legal Requirements

Whilst the legal requirements are usually dependent on the use case/business case followed by the company, once defined the fulfilment of these requirements is mostly solely dependent on the licence and the applicable laws.

No copyleft effect for add-ons or combinations

Determining if the licence has a copyleft effect is fairly easy. Yet, it should also be analysed when the copyleft effect applies to add-ons or combinations. In many licences there is a way of combining the F/OSS with other software and still avoiding a “licence contamination”.

No liability for third party code

Examining whether the company is liable for third party code when selling it, is determined solely by the criterion “liability when reselling”.

No patent infringements

No criterion can measure whether the F/OSS in question violates some software patent, but if the company does not have the budget to research the possible patent infringements within the F/OSS, a good indication is to find other users. If large companies use the software it is likely that these companies researched the software for patent infringements before using it, unless, of course, the company using it holds the patents itself.
6.5 **Fulfilment of Economical Requirements**

The following chapter discusses how the fulfilment of economical requirements can be estimated by interpreting the according criteria.

**Sustainability**

If there is a large, active and international community with reputable companies participating in the development and supporting the F/OSS it is most probable that sustainability is granted because this community doesn’t seem to be about to die. Base factors are satisfactory feature coverage and an adequate direction of the product evolution.

**Protection for investments**

Like the sustainability requirement and complemented by effort and cost considering criteria it can be stated that if stable development circumstances are given and the migration and running of the F/OSS can be expected to remain economical justifiable a sufficient protection for investments should be given because investment is not too high and the time span for amortizing the invest by using the product can be expected to be long enough.

**Increase productivity**

To increase productivity, it is important that a sufficient coverage of needed features is given and that the product can be recognized to evolve with further features useful for the company so the F/OSS remains productivity improving.

**Flexible maintainability according to individual needs**

Sufficient flexible maintainability is given if starting from a satisfactorily coverage of necessary features the available skills and resources for further development of the F/OSS seem to be enough and the open feature requests tend to be closed with adequate pace.

**Quick availability**

Based on an adequate feature coverage it is important to know the product dependencies and their possibly disturbing side effects to estimate if the product can be available quick enough.

**Increasing Know-how**

If the developer community seems to be skilled enough and the climate in the discussion forums is progressive and cooperatively it is most possible that the in house staff will be able to learn from the community and to increase its own Know-how.
Cost reduction

When sufficient feature coverage is given and the development tends in a helpful direction the basis for cost reduction is given. Further necessary are justifiable migration and running costs and a good documentation for the users to reduce support effort. Helpful for cost reduction are furthermore a minimum of product dependencies and a cooperative climate in the discussion forums to decrease maintenance effort.

Division of development costs

Division of development costs can be reached by feature coverage and the right evolution direction. Further necessary is a living and sufficient resourceful development community with stable participants. Existent documentation doesn’t have to be written in house and contributes to lower the development effort.

6.6 Fulfilment of Political Requirements

In the following chapter it is discussed how the fulfilment of political requirements can be estimated by interpreting according criteria.

Possibility for influencing further development with respect to individual needs

The possibility for influencing further development of an F/OSS with respect to individual needs can be estimated according to several criteria. If the F/OSS evolves in a satisfactory direction and sufficient development resources are given the F/OSS can be developed with respect to users needs. A minimum of product dependencies is helpful because this can be a crucial constraint for further development. The climate in the discussion forums shows how open the community is for change requests from product users. By the number of open feature requests in relation to time can be seen how efficient the development progresses.

Decrease of proprietary dependencies

A decrease of proprietary dependencies can be achieved, if the feature coverage of the F/OSS is sufficient and it evolves in a useful direction. If the community has enough development resources and skills available this is helpful to remain independent with the help of the F/OSS. As well it is useful if well reputed companies are involved and this relation seems to be stable in the long run.

Transparency over security
Additional to the insight of the source code it is helpful to gain transparency over security if the community is skilled enough to produce a readable design of the code and a can be gained the climate in the developer forums is cooperative. A minimum of dependencies from other products may increase the transparency further.

**Increase publicity, marketing effects**

Publicity and marketing effects by participating in an F/OSS community can be increased if the whole project is well known and reputable. This is often measured by well known companies and by the amount of developers, testers and users with sufficient development resources participating in an F/OSS project.

## 7 Summary and Conclusions

There are various business cases which can make the use of F/OSS within a company attractive. The problem is that often the companies have no experience in deciding when it is wise to use a specific F/OSS product or not.

This paper provides an approach based on decision criteria to support decision-makers in the judgement of integrating a specific package of F/OSS into the company portfolio, by analysing the environmental circumstances of the F/OSS project.

A decision to integrate a piece of software into the company is highly dependent on the intended use for the software. Therefore the paper starts by listing possible scenarios for adoption of F/OSS. Different requirements towards the F/OSS and its project environment are derived from and put into relation with the scenarios for adoption. For better lucidity the requirements are structured into the aspects functional, technical, organizational, legal, economical and political. This structure is pursued throughout the rest of the document.

The paper then shows different criteria which can be extracted out of the project environment and which provide useful information about the F/OSS. This information can later be used to determine if the posed requirements are met.

Finally the paper illustrates which criteria to use and how to use them in order to decide whether the requirements posed at the F/OSS are actually met.

It is very important to notice that the presented approach is not and cannot be understood as an automated decision system for the adoption of F/OSS products into a company. It is a guideline which eases the
choice through a systematic analysis of relevant factors. Every decision-maker has to adapt this approach according to the specific circumstances and individual preferences and needs.

### 8 References


