The FAMEtool: an automated supporting tool for assessing methodology

FILIPPO EROS PANI*, GIULIO CONCAS*, DANIELE SANNA**, LUCA CARROGU**

*Department of Electric and Electronic Engineering, Agile Group
University of Cagliari
Piazza d'Armi, 09123 Cagliari
ITALY
{filippo.pani, concas}@diee.unica.it http://agile.diee.unica.it

**FlossLab Srl
Viale Elmas 142, 09122 Cagliari
ITALY
{daniele.sanna, luca.carrogu}@flosslab.it http://www.flosslab.it

Abstract: The FAMEtool is a support tool for the FAME (Filter, Analyze, Measure and Evaluate) Methodology, an iterative approach for open source software assessment. The FAMEtool does not replace evaluation experts, but it serves to improve the productivity of experts in evaluating the different solutions. The efficacy of the FAMEtool is reported in this paper. Several methods have been created to define a process for assessing Free/Open Source software. Some focus on some aspects like the maturity, the durability and the strategy of the organization around the Open Source project itself. Other methodologies add functional aspects to the assessment process. Only some of these methodologies are supported by specific tools. This paper describes the FAMEtool that implements the steps of the FAME methodology, particularly in the filtering and analysis phases, trying to follow the logic of the simplified method that allows a faster application and gives support to who will be engaged in the selection of the new IT solution. Such a tool is introduced in the form of Web application and has been developed in Java.

Key-Words: Open Source software, Software evaluation, Technology transfer, Software quality, Assessment model.

1 Introduction

The choice of technologies for own IT investment is fundamental for organizations, because they influence practically all their businesses processes. Therefore, the optimal choice of their architecture and software components is of paramount importance. A wrong choice can lead to dire consequences and inefficiencies such as information loss, higher maintenance and redesign costs, stop of operational activities, and so on.

In recent years, Free/Open Source Software (F/OSS) emerged as a viable solution for software applications [1-7].

The increasing interest in F/OSS is patent in many different contexts like communities of individual users, private firms focusing their attention on this kind of approach, and public institutions.

The European Commission is currently funding several research projects related to Open Source and quality, namely, QUALOSS [8] FLOSSMetrics [9], SQO-OSS [10] and QUALIPSO [11].

It is very difficult to decide which F/OSS application to adopt inside an organization, because the number of Open Source projects is strongly increasing. Some products have their own web site as the main distribution mechanism for the software. However, the most F/OSS products are available through portals, which act as repositories of projects.

On SourceForge alone, one of the most important repositories, more than one hundred thousand projects are hosted. So the myriad of F/OSS products makes actual adoption a real challenge, and it is necessary to have methods to assess and compare these software products [12-17].

In the last few years, many F/OSS evaluation methodologies have been proposed to address this issue. All these methodologies have effectiveness as their main goal, and this goal leads to a hardly sustainable increase of their complexity, from both the perspective of their costs and needed competences. All evaluation frameworks reported in the literature were devised using an analytic research approach, trying to analyze many factors [18-22]. For this reason, such frameworks are often not easily applicable to real environments, especially in the case of small organizations.

This paper presents a new tool to support F/OSS maturity and reliability evaluation methodology, called FAMEtool for "Filter, Analyze, Measure and Evaluate". FAMEtool is aimed to support the use of FAME approach that reduces the evaluation complexity in order to be easily usable also by SMEs and small public bodies.

The FAMEtool has been developed to support the use of the FAME approach [23] that is a new light methodology derived from previous studies performed at the University of Cagliari that are characterized by a rigorous, but heavyweight approach to F/OSS selection [18][24][25].

The goals of FAME methodology are to aid the choice of high-quality F/OSS products, with high probability for sustainability in the long term, and to be as simple and user friendly as possible. The evaluation is not only about technical features of the product and quality of its development community, but it also takes into account a cost-benefit analysis specific of the involved organization.

The article is organized as follows. In Section 2 we briefly analyse the main existing assessing methodologies with its support tools. In Section 3 we describe the origin and the characteristic of the FAME approach and analyse its 4 phases. In Section 4 we present and discuss the FAMEtool. Section 5 concludes the paper.

2 Software Assessment Methodologies and Tools

Many studies [3][12][17] investigated whether the maturity of the processes employed by distributed, volunteer projects is linked to their success. The results of these studies clearly showed the importance of the maturity, the durability and the strategy of the organization around the Open Source project itself. Also the quality of code is a very important feature. These studies identified the importance of the use of version control tools, effective communication through the deployment of mailing lists, and found several effective strategies related to testing.

Some studies [26] also analyzed how Knowledge Management is important in Open Source teams because of the nature of the communities; these studies demonstrated that sharing knowledge and free access to information are fundamental for the development and the growth of these communities.

Regarding software development, we know that modern software systems can be composed by tens of thousands of different files, or modules. To verify the quality of software very complex approaches can be used; some studies [27] shows that the fractal dimension of software networks is a significant metric to describe the regularity of the software structure. These approaches need complex analysis and a recent study [28] analyzes performances with respect to different values of some parameters related to the Yule process associated to the preferential attachment. The assessment methodologies are neutral with respect to metrics, they can use complex or simple approaches.

Despite the widespread use of Open Source products in academic and industrial environments, only recently first attempts have been made to evaluate Open Source products. Some significant contributions are mentioned in this section. All these methodologies have a common criteria; they also present various phases and are based on scores.
Some among the main F/OSS evaluation methodologies are the following.

2.1 OpenBRR

OpenBRR (Open Business Readiness Rating) is being proposed as a new standard model for rating Open Source software. It is intended to enable the entire community (enterprise adopters and developers) to rate software in an open and standardized way [25].

The OpenBRR assessment has four phases, as shown in Fig.1:

1. performing a quick assessment to rule in or rule out software packages and to create a short list of viable candidates;
2. assessing the intended use for the software;
3. gathering and processing the relevant data;
4. translating the data into a numeric score from 0 to 5, where a high score represents a greater business readiness.

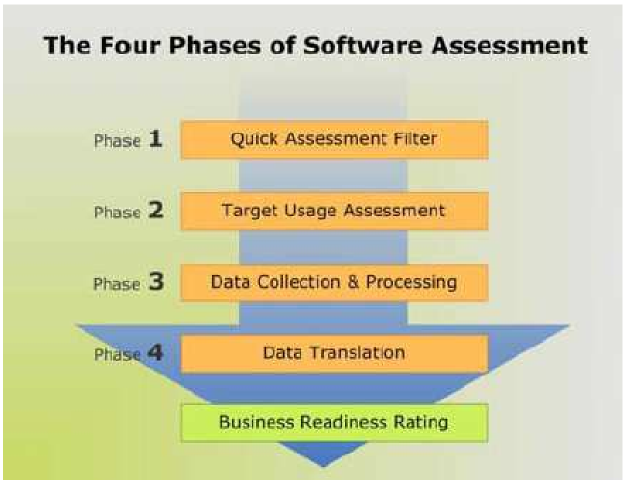

The Four Phases of Software Assessment
Phase 1
Quick Assessment Filter
Phase 2
Target Usage Assessment
Phase 3
Data Collection & Processing
Phase 4
Data Translation
Business Readiness Rating

Fig.1

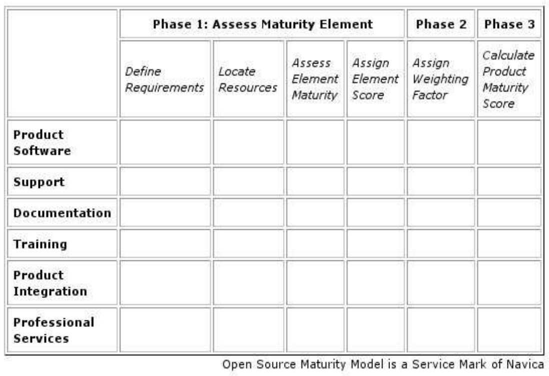
The assessment process is based on categories that are important for the Open Source evaluation process. They use those categories, along with those found in standard evaluation process documents (such as ISO/IEC 9126 and the newly released ISO/IEC 25000), and condensed them down to seven areas for evaluation:

1. Functionality;
2. Operational software characteristics;
3. Support and Service;
4. Documentation;
5. Software technology attributes;
6. Adoption;
7. Development process.

The Business Readiness Rating model is open and flexible, yet standardized. This allows for a broad implementation of a systematic and transparent assessment of both Open Source software and proprietary software.

2.2 The Open Source Maturity Model by Navica

The Open Source Maturity Model (OSMM) by Navica is designed to help organizations to successfully adopt and implement Open Source software. It consists of a three-phase process for selecting, assessing and implementing F/OSS products [29].



	Phase 1: Assess Maturity Element				Phase 2	Phase 3
	Define Requirements	Locate Resources	Assess Element Maturity	Assign Element Score	Assign Weighting Factor	Calculate Product Maturity Score
Product Software						
Support						
Documentation						
Training						
Product Integration						
Professional Services						

Open Source Maturity Model is a Service Mark of Navica

Fig.2

In Phase 1 of the OSMM, an organization evaluates each product element with a four-step process: define requirements, locate resources, assess element maturity, and assign element score. Based upon the organization's particular requirements, the available resources are assessed for their maturity and a score between 1 and 10 assigned. The output of Phase 1 is a set of scores for each of the key product elements. Of course, not every product element is of equal importance. Software is fundamental; support is critical; documentation, though necessary, is less important than the previous two elements.

In Phase 2 of the OSMM, weightings are applied to the individual element scores to reflect their overall importance for the product maturity. Default weightings are provided, but each organization is free to adjust the default weightings to reflect its particular needs.

An overall product maturity score is calculated in Phase 3 of the OSMM. This can be compared to the recommended minimum scores to determine if the product is suitable for an organizations needs.

The score can also be evaluated to determine if there are problems with the product that the organization needs to mitigate. The recommended minimum scores are, of course, just that: recommendations. The organization does not have to follow them rigidly; the recommended scores serve as guidelines to help to determine if an Open Source product will serve its needs. Using the key software concept of maturity (i.e., how far along a product is in the software lifecycle, which dictates what type of use may be made of the product), the OSMM assesses the maturity level of all key product elements:

- Software;
- Support;
- Documentation;

- Training;
- Product integration;
- Professional Services.

The output of an OSMM assessment is a numeric score between 0 and 100 that may be compared against recommended levels for different purposes, which vary according to whether an organization is an early adopter or a pragmatic user of IT.

2.3 The Open Source Maturity Model by Capgemini

In order to be able to determine if an Open Source product is suitable for an organization, Capgemini developed its Open Source Maturity Model (OSMM) [30]. The OSMM describes how a F/OSS product should be assessed to ensure that the product meets the IT challenges companies face today. The OSMM accomplishes this by linking an extensive product analysis with a thorough review of the company and its IT issues.

The OSMM aims to:

- determine the maturity of an Open Source product;
- access an Open Source product's match to the business requirements;
- compare Open Source products with commercial alternatives;
- show the importance of an Open Source Partner (OSP).

The products are compared using the product indicators, that are grouped into 4 different groups:

- Product;
- Integration;
- Use;
- Acceptance.

Each of these groups consists of a number of indicators, which together form the product score. The group "Product" focuses on the "internals" of the product, like the development and stability of the developer group or the purpose of the product. The group "Integration" measures the options to link the product to other products or infrastructure. In addition, it is also a measure for the product's modularity. The "Use" group tells us something about the way in which the user is supported in everyday use of the product.
For instance, by reviewing the number of support options made available to the user. The "Acceptance" group is all about the way the product is received by the user community, as this is largely indicative of the products ability to grow and become a prominent product. The indicators have a value between 1 and 5, 1 means not important (low) and 5 means extremely important (high).

2.4 QSOS

In order to have a method of qualification and selection of open software, Atos Origin built an original methodology to evaluate F/OSS software called QSOS [21]. The general process of QSOS is made up of several interdependent steps.

The goal of step 1 (Definition) is the constitution and enrichment of frames used in the following steps.

The frames of reference are :

- Software families: hierarchical classification of software domains and description of functional grids associated with each domain;
- Types of licenses: classification of free and Open Source licenses;
- Types of communities: classification of community organizations existing around a free or Open Source software and in charge of its life-cycle.

The objective of Evaluation (step 2) is to carry out the evaluation of the software. It consists of collecting information from the Open Source community, in order to:

- Build the identity card of the software
- Build the evaluation sheet of the software, by scoring criteria split on three major axis:
    - functional coverage;
    - risks from the user's perspective;
    - risks from the service provider's perspective.

The goal of Qualification is to define filters translating the needs and constraints related to the selection of Free or Open Source software into a specific context. This is achieved by "qualifying" the user's requirements which will be used later in the Selection phase.

The last step, Selection, identifies software fulfilling user's requirements, or more generally compares software from the same family.

2.5 EFFLOSS

This approach is intended to help IT organizations assessing which Open Source software would be most suitable for their needs [22]. The main limit of the frameworks described above is that they are based on qualitative metrics.

The key idea of EFFLOSS is to systematically use quantitative metrics that can be automated, taking advantage of the information that can be found on the Web.

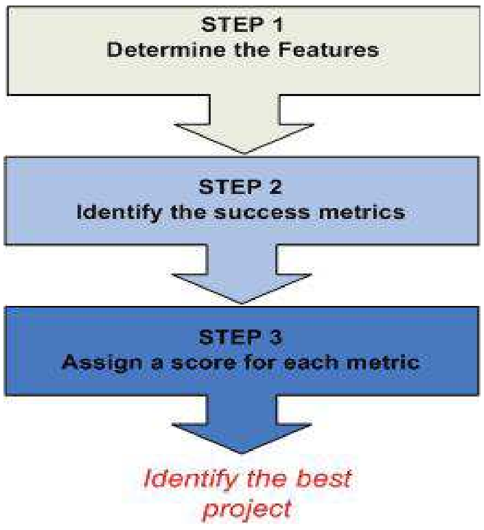

STEP 1
Determine the Features
STEP 2
Identify the success metrics
STEP 3
Assign a score for each metric
Identify the best project

Fig.3

In order to allow an easy usage of EFFLOSS, the assessment process is divided in three steps (see Fig.3), which are performed sequentially: the first one determines the features that characterize an Open Source product [31]; the second one identifies the success metrics; the third one assigns a score for each metric.

The goal of the first step is to understand in which ways a particular OS product is similar or different from another Open Source product. In this phase the authors define the feature that characterize and affect the success of Open Source products in order to perform a preliminary analysis.

The maturity of a product is key to understanding how well it is suited for a particular use. Relatively immature products can be used for noncritical systems, whereas production use require very mature products. Determining the maturity of a product is a critical priority for any IT organization, because the failure to assess a product’s maturity early enough in the selection process yields dire consequences. For commercial software products, pragmatic IT organizations expect a single company to deliver the elements required for sufficient maturity: the product, training, support, and so on. If the company does not deliver an element itself, it has recommended providers that will deliver that element at the required level of maturity.

2.6 NVAF

The NVAF is a framework that aims at giving a contribution to support the public administrations in their decision-making choices [24].

NVAF (Needs, Values and Assessment Framework) is addressed to public administrations and has a neutral approach to assure impartiality in the adoption choice of IT solution based on proprietary software or Open Source, and to avoid any kind of discrimination. The framework implements the criteria for evaluating the strategic choice of a solution compared to another. It allows evaluation if the choice increases the value into the environment in question.

The public administrations should evaluate which solution amongst those available is more suitable to their needs by comparing technical and economical factors, and also taking into account total own cost of individual solutions and cash outflows. The choice of the software model to adopt is a strategic decision for the public administrations and it should be based on the maximum value for money, which is defined like "the best combination of the cost of owning a system and the quality of the system, based on its capacity to satisfy requirements".

It is necessary to consider the investment in its totality and not in separate parts that are independent of one another. Moreover it is necessary to assure the PA’s adopted solution will guarantee the maximum value for money measured by the merit and local needs of business.

NVAF is a framework composed by three factors interacting among each other: Needs (N), Values (V) and the Assessment (A). These factors are the categories of the framework and the structure is supported by the key elements for the PAs that are identified as:

- actors and their roles;
- interest areas;
- processes.

NVAF aims to manage the complexity of this multitude of factors using a matrix to conceptualize the interrelation among the components and then to map them in order to understand the same framework.

3 FAME: Filter, Analyze, Measure and Evaluate Approach

FAME methodology originates from previously described heavyweight Open Source assessment methodologies. FAME can be considered an evolution of many software comparison methodologies and it focuses on some aspects like the maturity, the durability and the strategy of the organisation around the Open Source project itself [23]. FAME takes into account many aspects also present in NVAF [24], intended to support the choice of software applications. NVAF is mainly addressed to public administrations and has a neutral approach to assure impartiality in the adoption of IT solutions based on proprietary software or F/OSS. The framework considers also the social impact of the choice. Another very interesting feature of this methodology is the use of quantitative metrics gathered on the Internet about the projects to evaluate.

All these methodologies, however, are quite heavyweight approaches, suitable for large organizations or research studies, but difficult to adopt by SMEs. For this reason, in the context of a research project of FlossLab, an Italian SME, it has been decided to devise a new methodology to support SMEs in selecting F/OSS applications, using a simple, structured and tool-based approach, with support from the University of Cagliari.

The main idea behind FAME is that the users should evaluate which solution amongst those available is more suitable to their needs by comparing technical and economical factors, and also taking into account the total cost of individual solutions and cash outflows.

This principle is strictly related to a set of conditions to account for, its constraints, disadvantages and benefits. First, the goals of the project need to be defined, and the planning approach has to follow a strategic investment choice. In particular, it also considers all the positive effects registered in the area where the investment takes place. The required activities to obtain these results are:

1. identify and evaluate the main constraints and risks;
2. identify and evaluate the needs of the involved organization;
3. identify and prioritize the key objectives;
4. provide a priority framework.

The stakeholders with strategic information are considered as users of the framework. They are in charge to make changes and to approve the choices of a specific project.

The methodology is structured according to four distinct phases (Fig.4).

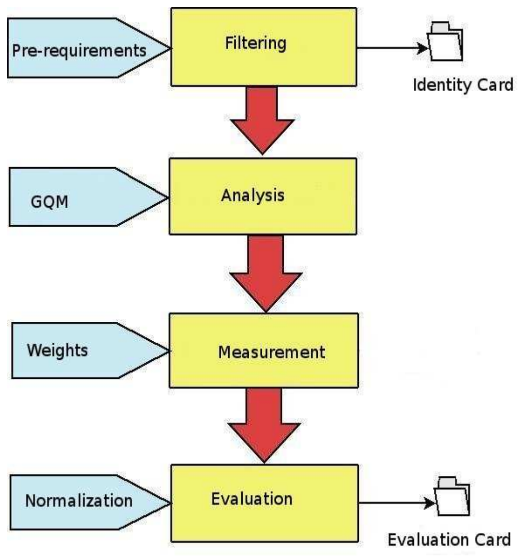


Fig.4

3.1 Filtering

The first problem to deal with in an evaluation methodology is the one connected to the choice of the candidate projects to introduce in the following assessment phases. In fact it is totally counterproductive to carry out any kind of evaluation on an excessive number of solutions.

From these considerations it follows before the real evaluation phase a selection phase is needed that reduces the number of options in a consistent way.

The first operation to complete in such process is therefore the choice of which are the projects that satisfy the minimum requirements (which does not come within a quality evaluation or however a detailed evaluation) connected to particular requirements of the organization that carries out the evaluation. This operation is of fundamental importance and must be as simple and fast as possible as it allows to reduce considerable and in an immediate way the solutions to take into consideration in the real phase of evaluation, with consequent cost reduction in terms of time and resources of the entire process.

The critical elements of the census operation can themselves be summarized in two main aspects: what to search for (domain understanding) and where to search (identification of possible repositories and citation of eventual studies).

Such information will allow to construct a general profile on the solution of our interest and therefore to define an Identity Card of the project by means of which we will be able to carry out a totally qualitative but highly effective evaluation for the filtering operations.

3.2 Analysis

From the user's point of view this phase aims to understand which solution can satisfy the needs of the organization considered in order to guarantee effective and efficient productivity.

About the needs analysis, the key is to seek the gap between the current situation and the desired situation and then to focus resources where they're most needed. The analysis must determine root causes.

The approach wants to give a correct and complete identification of the business objectives that we want to get, with the focus on the needs to satisfy. The output will be established by a hierarchy based on priority of objectives, obtained from strategic evaluations and the domain characteristic and constrains.

The process of needs definition has been divided in three main steps. The first step has involved an analysis of the literature and of the Italian and international regulation and guidelines. The second one has involved the creation of a survey for the decision makers according to the results of phase one. In the third step the stakeholders define the more interesting needs and its priority based on the needs knowledge base.

In the third step we can record the necessity that the survey has to have some open areas to identify other potential needs not yet found or specific of the organization. The complete identification of the survey areas is still on and can be modified.

The surveys have been organized in two macro-areas, they recognize the main type of needs. Therefore the final components, based on the information are the following:

- technical and functional analysis;
- economical and social analysis.

3.3 Measurement

The measurable elements come from the needs. FAME turns the high level stakeholder's evaluations in technical-functional values and economical-social values. A weight and a metric is associated to these elements and the comparative analysis among the solutions is possible.

The technical-functional elements are classic in literature, and so we show only the economical-social elements concerning the evaluation of potential benefits for the citizens, the enterprises and local organizations. They take into consideration the social and educational elements, like the offer to wide access to information, the increase of capacity and information skills of the citizens. But it is very important to consider all business implications around the territory, like the increase of local capacity and skills with important repercussions on the development of the local enterprises.

We have adopted the procedures of the economical analysis following the cost-benefit analysis. This analysis is very difficult and expensive to follow in-depth way, so we have adopted only the methodological approach. This approach is used to make firstly an estimate of benefits, secondly of costs, then both, also other intangibles. Because a cost-benefit model shows if the system benefits justify your implementation, so it is necessary first to see the value coming from the adoption of a solution rather than another. This analysis aims to find the typical real costs of a project choice and to evaluate and compare incidental saving costs for the public administrations, so the public administrations with these savings can offer further services. A correct procedure of an economic comparison should be completed both with the starting costs and the services costs to the support, the training, but also migration, installation and management costs, adapting, maintaining, and so on.

The costs compared to benefits are simpler to find. The right costs analysis should also take into account the TCO (Total Cost of Ownership). It considers all direct and indirect costs. All software has a TCO including the price of selling, hardware and software upgrades, maintenance, technical support and learning (time and frustration are complex to measure). After we have assigned a cost to each single item, this cost will be normalized. Here we suggest a possible solution and how our normalized score is calculated:

$$S = W*(Cmin + Cmax - Cij)/Cmax$$

let: *[S]* normalized score; *[W]* maximum assignable score; *[C_min]* lower price; *[Cmax]* higher price; *[Cij]* price to be normalized.

FAME associates to the need of type outsourcing the technical-functional elements of type "supplier reliability" with high weight. This is because the organization, which we have analyzed, assigned a high priority to that need. FAME relates to each Need being in the questionnaire a set of Measurable elements with corresponding metrics, so that to each element *Ei* is related an objective metrics of evaluation *Mi*.

3.4 Evaluation

The decision choice should be taken by comparing the values of the needs among the different solutions, and using the weights and the objective metrics for the assessment of the found elements. In order to compare different objects, we choose to adopt a systematic comparison among the scores of the solutions.

The approach is based on associating a weight to a value, it reflects the relative weight of the value in the overall assessment in accordance with $\sum Wi=Wtot$, where *Wi* is the weight associated with the element *Ei* and *Wi /Wtot* is the relative weight in the assessment based on the importance coming from the needs analysis and the priority that one has associated. The priority comes from the evaluation given in the questionnaire to the need from which the value has come.

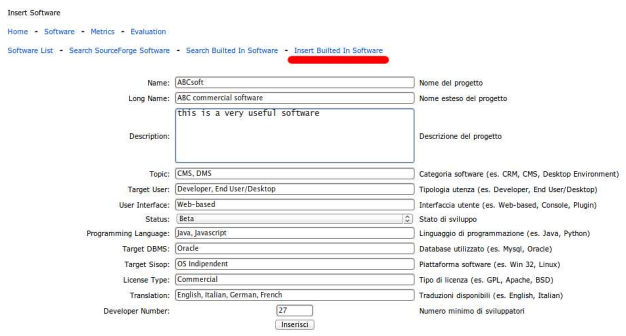
From this phase FAME obtains the table that is instantiated with the analyzed need and it shows also how the weights depend on the needs. Then we compare the different eligible solutions. The project choice will be determined throughout the metrics *M* used with the weights *W* associated with the element *E*, in accordance with FAME. Each solution will have a final score like summation of the scores assigned with each evaluated element. Finally the organization will choose the project solution with the major score.

Practically, a score *Pij* is assigned to each proposed solution *Sj*, where $0<=Pij <=Wi$ for each element *Ei* based on metric Mi associated with that element. Then each solution will have a total score *Pjtot*. The solution with the major score *Pjtot* will be the chosen solution, in fact it will satisfy better the needs of the administration. The solution with the maximum *Ptot* will be the best because the evaluation comes from the needs analysis and elements measurement and the assessment of that organization.

4 FAMEtool, description and use

Purpose of this support tool is to supply useful instruments in order to simplify and automate as much as possible the application of a determined evaluation methodology. The tool is of easy application also for non professional people and is like a guide for the user who will have to carry out the selection among various software alternatives and it will assist him in the evaluation phase.

FAMEtool implements the steps of the FAME framework, trying to follow the logic of the simplified method and proposing automation that allow a faster application and supplies support to those who will be engaged in the selection of the new IT solution. Such tool is introduced in the form of Web application and has been developed in Java.



Insert Software

Home - Software - Metrics - Evaluation

Software List - Search SourceForge Software - Search Builted In Software - Insert Builted In Software

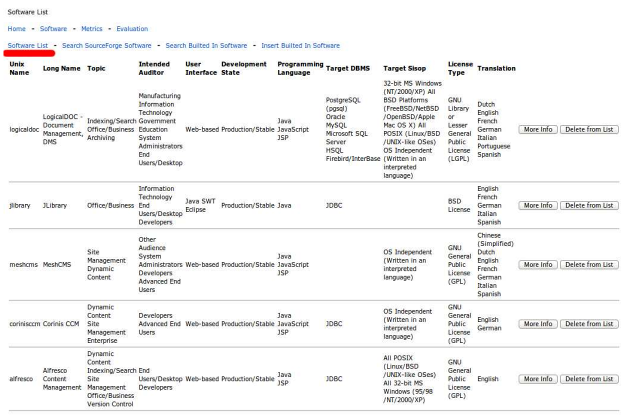
Name:	ABCsoft	Nome del progetto
Long Name:	ABC commercial software	Nome esteso del progetto
Description:	this is a very useful software	Descrizione del progetto
Topic:	CMS, DMS	Categoria software (es. CRM, CMS, Desktop Environment)
Target User:	Developer, End User/Desktop	Tipologia utenza (es. Developer, End User/Desktop)
User Interface:	Web-based	Interfaccia utente (es. Web-based, Console, Plugin)
Status:	Beta	Stato di sviluppo
Programming Language:	Java, Javascript	Linguaggio di programmazione (es. Java, Python)
Target DBMS:	Oracle	Database utilizzato (es. Mysql, Oracle)
Target Sisop:	OS Indipendent	Piattaforma software (es. Win 32, Linux)
License Type:	Commercial	Tipo di licenza (es. GPL, Apache, BSD)
Translation:	English, Italian, German, French	Traduzioni disponibili (es. English, Italian)
Developer Number:	27	Numero minimo di sviluppatori

Inserisci

Fig.5

In the Filtering phase the system supports the user in the collection of information about the candidate products through an user interface of data insertion, shown in fig. 5, and a module of data imported from repositories of Open Source projects, like for example SourceForge. All the information and data gathered are reclassified and will serve as a base in order to be able to search software on the base of functional requirements needed.

One of the strong points of this kind of approach is that the user can define and insert through an iterative filtering process, further characteristics with relative values and in this way select further candidate projects, all through a comfortable interface of selection.



Software List

Home - Software - Metrics - Evaluation

Software List - Search SourceForge Software - Search Builted In Software - Insert Builted In Software

Unix Name	Long Name	Topic	Intended Auditor	User Interface	Development State	Programming Language	Target DBMS	Target Sisop	License Type	Translation		
logicaldoc	LogicalDOC - Document Management, DMS	Indexing/Search Office/Business Archiving	Manufacturing Information Technology Government Education System Administrators End Users/Desktop	Web-based	Production/Stable	Java JavaScript JSP	PostgreSQL (pgsql) Oracle MySQL Microsoft SQL Server HSQL Firebird/InterBase	32-bit MS Windows (NT/2000/XP) All BSD Platforms (FreeBSD/NetBSD /OpenBSD/Apple Mac OS X) All POSIX (Linux/BSD /UNIX-like OSes) OS Independent (Written in an interpreted language)	GNU Library or Lesser General Public License (LGPL)	Dutch English French German Italian Portuguese Spanish	More Info	Delete from List
jlibrary	JLibrary	Office/Business	Information Technology End Users/Desktop Developers	Java SWT Eclipse	Production/Stable	Java	JDBC		BSD License	English French German Italian Spanish	More Info	Delete from List
meshcms	MeshCMS	Site Management Dynamic Content	Other Audience System Administrators Developers Advanced End Users	Web-based	Production/Stable	Java JavaScript JSP		OS Independent (Written in an interpreted language)	GNU General Public License (GPL)	Chinese (Simplified) Dutch English French German Italian Spanish	More Info	Delete from List
corinisccm	Corinis CCM	Dynamic Content Site Management Enterprise	Developers Advanced End Users	Web-based	Production/Stable	Java JavaScript JSP	JDBC	OS Independent (Written in an interpreted language)	GNU General Public License (GPL)	English German	More Info	Delete from List
alfresco	Alfresco Content Management	Dynamic Content Indexing/Search Site Management Office/Business Version Control	End Users/Desktop Developers	Web-based	Production/Stable	Java JSP	JDBC	All POSIX (Linux/BSD /UNIX-like OSes) All 32-bit MS Windows (95/98 /NT/2000/XP)	GNU General Public License (GPL)	English	More Info	Delete from List

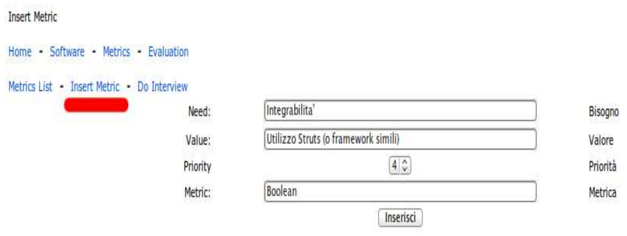
Fig.6

At the end of this phase the FAMEtool shows, Fig. 6, the list of candidate projects and the user can view more info about a project or delete it from the candidate list.

In the Analysis phase the user is guided through a tree in which the leaves represent the needs to measure and the nodes the several macro area of such needs. This tree is visited only if the interest manifested by the interviewed user for the macro area will be above a defined threshold; in the contrary case, all the macro area will be neglected.

During this iterative process the user chooses and defines indirectly the metrics to evaluate through the manifestation of the needs. Inside the tool an interactive questionnaire has been constructed: every question has been formulated in order to be able to directly find the priority of the need to which it is associated. The possible answers proposed for every question go from “Not important” to “Very important” and they are numerically translated in a value comprised between 1 and 5.

Because it's not possible to take into account through just one questionnaire all the multiple aspects that characterize a specific context in the choice of a new IT solution, it has been thought useful to be able to insert manually the tern Need, Value and Metrics, with priority, according the exigencies. In this way it is possible for the organization to accord the framework to its own needs keeping the evaluation phase open and dynamic.


Insert Metric
Home • Software • Metrics • Evaluation
Metrics List • Insert Metric • Do Interview
Inserisci

Need:	Integrabilita'	Bisogno
Value:	Utilizzo Struts (o framework simili)	Valore
Priority	4	Priorità
Metric:	Boolean	Metrica

Fig.7

In the Measurement phase the user is guided, as shown in fig. 7, through a wide view of what has to be evaluated and with what parameters such process must be executed. In literature there exists techniques defined in order to collect the necessary data for the evaluation metrics and they take the name of Evaluation Depth Classes (EDC) [28].

These classes can be divided into the following groups:

- documentation, interview, clarification;
- expert knowledge;
- non functional testing;
- scenario-based testing;
- prototyping.

The efforts necessary to collect the data belonging to each class grows gradually from the first to the last class; starting here, it's realized that there are metrics easily measurable and others less easily measurable, depending on if a data belongs to a certain class.

In some cases it appears impossible to identify the necessary information or onerous to start the activities for their collection. For example, let's think about the activity tests that require such effort for which the importance of the produced data is not justified.

Users could, therefore, decide to not carry out some measurements, discarding in this way the estimation of the associated need.

Finally, in the Evaluation phase the collected data will have to be inserted, after normalization, in an appropriate mask created on purpose by means of the list of the candidate software and the list of the metrics selected in the previous steps; the application will calculate the result taking into account the expressed priorities. The tool offers the possibility to save and, successively, reload, the entire data set used in the evaluation cycle; this data set includes the list of the candidate software, the list of the selected metrics with relation to the needs and obviously the obtained scores from every product, with graphs in order to have a wide view and an immediate interpretation of the obtained result.

4.1 Experience of use

To define FAME and its tool we use as test case a real need of a SME, which decided to enter the Document Management System (DMS) market with a F/OSS approach. To this purpose, they had to choose the Open Source DMS best suited to their needs. The skills of the firm were mainly in Java. In the following, we will briefly describe how the four phases of FAME were applied to this specific case with the assistance of the FAMETool.

In the Filter Phase, key prerequisites of the programs to choose are selected, with the goal to reduce the number of potential candidates for the next phase. These pre-requisites can be linked to strategic issues, such as quality and sustainability of the project, and to technical issues, such as the target operating system, programming language, and so on.

With the support of the tool, we have recovered the relative information of a high number of projects on SourceForge. FAMEtool also supported the insertion of the relative information of other interesting projects not classified by SourceForce as DMS (e.g. Alfresco).

In our case, we set a filter to restrict the DMS projects to those written in Java and PHP, because of their portability. Another filter was set to consider only highly popular – and hence highly downloaded and with a high activity index – projects. Since Java is the language used by the SME, we set a preference towards Java by lowering the latter filter requirements in the case of Java projects with respect to PHP ones.

After this filtering, we came out with a list of four projects, namely Alfresco, e-prot, KnowledgeTree and Nuxeo. Among these, only KnowledgeTree is a PHP project, all others being Java ones.

We also allowed for two more projects: Adam, an Italian DMS promoted by Italian Center for Open Source in Public Administrations (CNIPA), and Hummingbird, a proprietary project evaluated for comparison purposes. The last part of this phase was to create an "identity card" for all pre-selected projects, gathering the most relevant information about them, as found in the Internet.

The Analysis Phase was the most critical and effort-prone. It consisted in eliciting the key needs of the SME, as regards the DMS to choose. Following FAME guide-lines, we performed an analysis of technical-functional and economic requirements.

We set high priority to the programming language, security (using popular protocols, including LDAP), quality of documentation, and community "strength".

In the Measurement Phase, FAMEtool allows to evaluate the various identified metrics using different units, depending on their specificity. Some of them were simply boolean (yes or no, corresponding to 1 and 0 respectively), denoting the presence or absence of a specific feature. Some had a score between 1 and 5, while others had a three-valued score (0, 1, 2). For the sake of simplicity, the relative weights were chosen identical to the relevance factor for the relative need, that is between 1 and 5.

The metrics were then applied on the relevant features of the pre-selected projects, using data found in their "identity card", SourceForge and projects Web sites and source code repositories, yielding a set of scores for each candidate product.

In the Evaluate Phase, the Tool normalized the measured values in such a way that it became consistent with each other, and to reflect comparable values in economic terms. Then, using the choices made in the third phase, it weighted them according to the relevance of the respective needs, obtaining the total scores for each area (technical and economic) and finally showing the total score of each solution. The DMS with the highest score was Alfresco, in both the technical and economic areas, and it was therefore the chosen project.

5 Conclusion

In this paper we presented the FAMETool, the support tool for the FAME approach that is a simplified methodology for F/OSS project assessment.

FAME is an engineering process structured in 4 phases full-defined that can be easily automated. In fact in those phases where it is important and necessary to evaluate a very large amount of data, like in gathering, analysis and measurement steps, it is possible to achieve the best benefits from the automated tool.

This tool also allows to carry out the whole process driving the user along the phases and in their application, and acting as system support in the use of the methodology.

The main characteristics of the FAMEtool are:

- the supported iterative approach;
- an automated filtering phase to pre-select project candidates, with automatic collection of data, easing the whole evaluation process;
- a specific interface to support the analysis of the needs of the organization performed using GQM approach and in practice being able to configure the evaluation process according to the actual organization's needs, both technical and economic;
- an engine that explicitly considers various viable metrics, including possible quantitative measurements of software repositories and Web hits related to the projects;
- the evaluation/comparison component that blends together different metrics, making them comparable and consistent, and performs the final evaluation.

In particular, the integration and the contextual development of the tool with the methodology defines this methodology with the precise objective to support and to simplify the evaluation, rendering almost automatically and instantaneously all the phases of the evaluation process, through iterative and interactive process for the user, trying to reduce in this way the complexity of the methodologies analysed, which have as weak points the high number of information to manage.

References:

[1] Feller, J., Fitzgerald, B., Hissam, S. and Lakhani, K., Perspectives on Free and Open Source Software, *MIT Press,* Cambridge, MA, 2005.

[2] Free/Libre and Open Source Software: Survey and Study: Final Report", 2003, available at: http://FLOSS.infonomics.nl/

[3] Senyard, A. and Michlmayr, M., How to have a successful free software project., *APSEC, IEEE Computer Society*, pp. 84–91, 2004.

[4] Antoniades, I. P., Stamelos, I., Angelis, L. and Bleris, G. L., A novel simulation model for the development process of open source software projects, *International Journal of Software Projects: Improvement and Practice (SPIP), special issue on Software Process Simulation and Modeling*, 2003.

[5] Feller, J. and Fitzgerald, B., A framework analysis of the open source software development paradigm, *ICIS 2000*, pp. 58–69.

[6] Gonzalez-Barahona, J. M., Pérez, M. A. O., Quiros, P. d. l. H., Gonzalez, J. C. and Olivera, V. M., Counting potatoes: the Size of Debian 2.2, *Upgrade - The European Online Magazine for the IT Professional*, Vol. II, No. 6, December 2001, pp. 61-67.

[7] Mockus, A., Fielding, R. T. and Herbsleb, J., A case study of open source software development: the Apache server, *ICSE '00: Proceedings of the 22nd international conference on Software engineering, ACM Press*, New York, NY, USA, 2000, pp. 263-272.

[8] QUALOSS, http://www.qualoss.org, 2008.

[9] FLOSSMetrics, http://flossmetrics.org, 2008

[10] SQO-OSS, http://www.sqo-oss.eu, 2008.

[11] QUALIPSO, http://www.qualipso.org, 2008.

[12] Michlmayr, M., Software process maturity and the success of free software projects, in Zielinski, K. and Szmuc, T. (Eds.), *Software Engineering: Evolution and Emerging Technologies*, *IOS Press*, 2005, pp. 3-14.

[13] Stewart, K. J. and Ammeter, T., An exploration study of factors influencing the level of vitality and popularity of open source projects, in. Applegate, R. L and De Gross, J. I. (Eds.), *Proceedings of the Twenty-Third International Conference on Information Systems*, 2002, pp. 853-857.

[14] Weiss, D., A large crawl and quantitative analysis of open source projects hosted on sourceforge, *Research Report RA-001/05,* Institute of Computing Science, Poznań University of Technology, Poland, 2005.

[15] Weiss, D., Measuring success of open source projects using web search engines, *OSS2005, Proceedings of the The First International Conference on Open Source Systems*, 2005, pp. 93-99.

[16] Crowston, K., Annabi, H. and Howison, J., Defining open source software project success, *International Conference on Information Systems* (*ICIS)*, 2003, pp. 327–340.

[17] Crowston, K., Annabi, H., Howison, J. and Masango, C., Towards a Portfolio of FLOSS project success measures, in Workshop on Open Source Software Engineering, International Conference on Software Engineering, Edinburgh, Scotland, UK, 2004. From http://flosspapers.org/180

[18] Cau, A., Concas, G. and Marchesi, M., Extending OpenBRR with automated metrics to measure object oriented open source project success, *The Workshop on Evaluation Frameworks for Open Source Software,* collocated in *The Second International Conference on Open Source Systems*, 2006.

[19] Ciolkowski, M. and Soto, M., Towards a Comprehensive Approach for Assessing Open Source Projects, in *Proceedings of the international Conferences on Software Process and Product Measurement* (Munich, Germany), Lecture Notes In Computer Science, SpringerVerlag, Berlin, Heidelberg, Vol. 5338, 2008, pp. 316-330.

[20] Deprez, J. C., Alexandre, S., Comparing assessment methodologies for free/open source software: OpenBRR & QSOS, *Lecture Notes in Computer Science*, Springer, 2008.

[21] Method for Qualification and Selection of Open Source software (QSOS), version 1.6, Atos Origin, 2006, available at: http://qsos.org

[22] Cau, A., EFFLOSS: An Evaluation Framework for Free/Libre Open Sourc*e*, PhD Thesis, 2007.

[23] Pani, F. E., Concas, G., Sanna, D., Carrogu, L., The FAME Approach: an assessing methodology, *WSEAS International Conferences*, Catania (Sicily), Italy, May 29-31, 2010.

[24] Mannaro, K., Concas, G., Marchesi, M., NVAF: un Framework per una valutazione di tipo comparativo delle soluzioni software nelle Pubbliche Amministrazioni, *The Second International Conference on Open Source Systems*, Esperta Workshop, Como, Italy, 2006.

[25] Business Readiness Rating, A Proposed Open Standard to Facilitate Assessment and Adoption of Open Source Software, 2005, available at http://www.openbrr.org

[26] Porruvecchio, G., Uras, S., Concas, G., Knowledge Management Aspects in Open Source Communities, *WSEAS International Conferences*, Catania (Sicily), Italy, May 29-31, 2010.

[27] Locci, M., Concas, G., Turnu, I., Computing the Fractal Dimension of Software Networks, *9th WSEAS Int. Conf. on Applied Computer Science* (ACS'09), Genova, Italy, 2009.

[28] Tonelli, R., Concas, G., Locci, M., Three Efficient Algorithms for Implementing the Preferential Attachment Mechanism in Yule-Simon Stochastic Process, *WSEAS Transactions on Information Science & Applications*, Vol. 7, 2010.

[29] Golden, B., *Succeeding with Open Source*, Addison-Wesley Professional, 2004.

[30] Duijnhouwer, F. W., Widdows, C., *Open Source Maturity Model* (Capgemini), 2003.

[31] Jones, C., *Software Assessments, Benchmarks, and Best Practices*, Addison-Wesley Information Technology Series, 2000.

[32] Ochs, M., Pfahl, D., Chrobok-Diening, G., Nothhelfer-Kolb, B., A method for efficient measurement-based COTS assessment and selection method description and evaluation results, *Fraunhofer Inst. for Exp. Software Eng.*, 2001.