Developing free software in companies

Amadeu Albós Raya

PID_00145047

Universitat Oberta de Catalunya
www.uoc.edu

Index

Introduction

In this module, we delve into the world of free software production and its most relevant features for the product, company and user community.

To begin with, we discuss the development of free software from the point of view of the project, considering the main aspects affecting the population and management of the project, and the participation of the user community in a variety of aspects.

The free software project formalises the relationship between the company and the user community. The adaptation of the specific features of this relationship is essential to achieving the aims of the project.

We then move on to describe the specific features of the free software user community and its management by the company. This management complements the production methodology and implements the relational strategy discussed earlier.

Finally, the module concludes with a case study of a real company that produces free software.

This module is structured as a guide for external reading, the aim of which is to provide more detail on the features of the various aspects that emerge and which are relevant to free software business production.

Objectives

After completing this module, students should have achieved the following aims:

1. To be familiar with the methodology of free software production.

2. To understand the importance of the user community for the development of products based on free software.

3. To identify and analyse the relevant factors affecting the success of free software production.

4. To understand the importance of formalising a methodology to complement the efforts of the company and the user community.

5. To obtain a deeper understanding of the direct and indirect implications of carrying out a development project based on free software.

1. Free software production

In this first section, we will focus on the production of free software from the perspective of its development or creation, i.e. without considering the possible business models that exploit it for profit.

Several subjects of this Master's degree, particularly those on software production[1], discuss the technological process characterising free software development at length.

(1)Introduction to software development, Software engineering in free software environments and Advanced concepts of software development.

This technological process supplements the methodologies allowing us to formalise a viable cooperative project that will last over time. In this sense, the cooperation of the user community on the free software project is crucial for obtaining a critical mass of users to enable the project to be viable.

Consequently, many of these methodologies and actions are designed to offer support and guarantees to relations between the project and the user community. To understand the importance of this relationship, we can simply visit the resources offered by the more popular free software projects to the user community.

Popular projects

For example, OpenOffice.org (http://contributing.openoffice.org/) and Mozilla (http://www.mozilla.org/contribute/).

To develop these concepts, over the next few sections we will describe three complementary points of view. First of all, we will consider some basic ideas on free software production. We will then briefly detail the main steps to take to implement a project based on free software. Lastly, we will detail the main aspects of free software project management.

1.1. Free software production

The production of free software, like the production of any software, responds to the need to solve a specific[2] technology problem.

(2) For example, to add functionality to an application or to troubleshoot malfunctions.

Although the technological process of refining and developing a free software application may share many similarities with an application based on proprietary software, the difference marked by the openness of the model gives it a special type of operation. In other words, the open and cooperative nature of its production affects the structure of quantitative and qualitative evolution down the versions.

Many authors have written about the specifics of producing free software. Since it is not the aim of this module to detail or describe these features at length, given that they are comprehensively dealt with in other subjects, we will focus here on pointing out some of the more interesting ones in our case.

To do so, we will consider some of the concepts in Eric S. Raymond's paper *The Cathedral and the Bazaar*, which analyses the special features of free software, particularly GNU/Linux.

Recommended website

E. Raymond (1997). *The cathedral and the bazaar* (http://www.catb.org/~esr/writings/cathedral-bazaar/).

- **The origin of production**
  Broadly speaking, the production of free software emerges from the particular needs of the user or developer in his or her daily activity. In other words, collaboration on the development of the software begins when we look for and find a problem that we want or need to resolve.

Early stages of production

The bulk of the foundations of free software are based on the publication of specific adaptations or developments made by workers in the performance of their daily work.

- **The user community**
  The free software user community, which includes both end users and developers and programmers, is the pillar that gives meaning to the definition of free software development.
  Treating users as partners in the production project is the easiest way to debug and improve the code quickly (if the collaborator base is big enough). Thus, collaborators are one of the most valuable resources for the development of the application, so it is also helpful to recognise good ideas and the solutions they provide.

- **Versions of the application**
  One of the features of free software production is the reuse and rewriting of the original code to create a new code that is either error-free or which has new features or improved performance (among other aspects).
  Moreover, free software development projects encourage the quick and regular release of the code, which means that the project activity is dynamic and continuous.

- **Coordination of production**
  The individual – or individuals – who coordinate/s the project must be able to manage the global potential of the user community, guiding the project's evolution without coercion and taking advantage of the resources and synergies offered by networks such as the Internet.
  The legacy of the application's code and coordination management are important for the future of the free software development project. The choice of a successor to control and manage production should not be left to chance.

1.2. The free software project

In addition to the technological and functional considerations of applications based on free software, one of the primary aims of any free software project is to disseminate the application or obtain a critical mass of users.

To put it another way, it is not very helpful for the future of the project if the generated code is not known and applied by potential users, even if specific problems or shortcomings have been addressed. This is also a necessary aim for its subsequent maintenance and evolution over time. In the case of free software, fulfilment of this aspect is essential for the creation of a stable and lasting user community.

Several guides have been written that, to a greater or lesser extent, contribute the necessary concepts for the creation and management of projects based on free software. In this section, we will develop this issue using Benjamin Mako's *Free Software Project Management HOW TO* article, which reviews the main features of the project from a practical angle.

Recommended website

B. Mako (2001). *Free Software Project Management HOW TO* (http://mako.cc/projects/howto).

Launch

Before launching a project based on free software, it is very important to design a solid structure that will withstand the subsequent development process with sufficient guarantees.

In general, the basic structure of the project must take into account the following:

- The need to create a new project, either with its own ideas and aims or through existing, related projects.
- The definition of the main features of the application (functionality, licensing, numbering, etc).
- The basic infrastructure to support dissemination of the new project and collaboration on its development (website, contact e-mail, etc).

Developers

Once the project has been launched, the next aim will be the integration and consolidation of the users and developers of the application. We must create policies and strategies allowing us to define and structure the collaboration of the latter.

Cooperation policies must meet two main aims:

- The coordination of internal and external production, including the delegation of responsibilities and protocols of acceptance for contributions.

- Production management, such as the structure of development branches and their associated repositories.

Users

With products based on free software, the users are often developers too (and vice versa). One of the main aims to take into account then are application tests, be they functional, operational, quality, etc.

Support infrastructure

The daily activity of a project based on free software could not be carried out without a support infrastructure adapted to its cooperative aims.

The key actions in this regard are carried out during the project launch. However, once it is up and running, we will need to adapt, improve and supplement the existing resources in line with the progress of the project.

Usual resources

Some of the most common resources in free software projects are: documentation, mailing lists, bug tracking systems and versions, forums, chats, wikis, etc.

The application

Undoubtedly the most important component of the project is the application, on which the rest of the aspects considered are based. One of the key features required by an application is for the user to have sufficient guarantees on the performance of each version released.

The release of versions is a sensitive issue that requires careful thought. Broadly speaking, we need to consider the following:

- Control of revisions for functionality and debugging (alpha and beta versions, candidate distribution, etc).

- When to launch the full version, i.e. when the code will be ready to offer guarantees that we and the users expect.

- How to release the version (packaged, source code, binary, etc).

Dissemination of the project

Lastly, as we initially explained, raising awareness of the project is important, but this task should be carried out taking into account whether we will want to reinforce its foundations over time.

As the project progresses, we need to think about publicising it in free software mailing lists or on *Usenet*, including the project in other public portals (such as *Freshmeat* or *SourceForge*), or advertising new versions of the application in the project's own mailing lists.

1.3. Project management

In this section, we will describe in detail the aspects of project management that, as founders of the same, we must keep in mind to guarantee success.

The concepts we describe in this section supplement those of the above sections, since they allow us to specify and improve the various actions considered. Hence, it is possible to find direct and indirect coincidences with these arguments.

To indicate the basics of the management of projects based on free software, we will take into account the considerations set down in Karl Fogel's *Producing Open Source Software*, particularly Chapter 5, entitled "Money".

Recommended website

K. Fogel (2005). *Producing Open Source Software: How to Run a Successful Free Software Project* (Chapter 5 "Money"). (http://producingoss.com/en/money.html).

Funding

The special features of free software projects mean that many contributions are informally subsidised (for example, when a company employee publishes the adaptations it has made to the code during his/her daily activities).

Donations and grants are also made, contributing direct income to keep the project going, but we must take into account the management of these funds, since much of the support afforded to a free software project is based on the credibility of its participants.

Types of participation

There are many types – and possible combinations – of financial participation in a free software project. This funding model also influences aspects that depend not only on the project but also on its environment and context of action.

Broadly speaking, participation in a free software project is related to the collaboration of its participants, the business model exploited by the company that promotes it (where applicable), the marketing activities undertaken, the licensing of the products involved and the donations made.

Open-ended contracts

The application's team of developers is very important for the development of the project and its future evolution. The stability and permanence of the participants in their posts of responsibility will strengthen the foundations and credibility of the project vis-à-vis the user community.

A stable project

Credibility is essential for all actors directly or indirectly involved in the project, since this cannot be transferred to substitutes. Moreover, loss of credibility can affect the future of the application and the project to varying degrees, so we need to take the appropriate measures to actively monitor and manage the project.

Decentralisation

One of the most relevant – and desirable – features of free software user communities is the distribution and decentralisation of the decisions taken in the project.

Hence, the project organisation should consider this structure as a way to motivate and strengthen the community of application users, ensuring that the consensus emerges from interaction between its members.

Transparency

The above aspect of decentralisation gives us an idea as to the transparency and justification that should exist in the relationship between the project and the community.

Both the aims of the project and lines of evolution of the application must be clear and well known to all those involved in it. The influence of the founder on future behaviour must be exercised in a sincere and transparent way in order to guarantee the credibility of the project[3].

(3)One example is the Openbravo manifesto (http://www.openbravo.com/es/about-us/openbravo-manifesto/).

Credibility

Project credibility (both overall and of its individual members) has cropped up in a number of the issues we have already discussed. Its relevance is closely related to the free software user community and it is an important prerequisite for maintenance of the project over time.

Money or a hierarchical position cannot generate the necessary credibility in the actions of individual members at any given time. In other words, the established methodology, procedures or protocols, or the workings or operation must be the same for everybody, without exception.

Contracts

Employee hiring is another aspect to take into account, particularly in free software projects, due to its impact on structure and operation. We need to ensure that all of the details and processes of recruitment are open and transparent.

In fact, it is important to review and approve these changes with the collaboration of the user community, to the extent that, in some cases, it may be preferable or desirable to contract developers directly from the community with write permissions on the official repository (committers).

Resources

Free software projects are based not only on the evolution and maintenance of the code of an application based on free software; they must also consider additional aspects of support.

Additional resources

This is the case of the quality management of the code produced, the legal protection of contributions, the documentation and utility of the application, and the provision of infrastructure resources for the free software community (websites, version control systems, etc).

These resources can generate significant differences in the dissemination and popularisation both of the application and of the project in the free software user community.

Marketing

Lastly, although we are dealing with a project based on free software, we should implement marketing measures for the dissemination and popularisation of the application and of the project as a whole.

Hence, we must remember that the full workings of the project are in the public eye and that each of the claims made may be easily demonstrated or proved wrong. The establishment of measures to control the image and operation of the project must enable it to gain credibility, transparency and verifiability.

These measures include the importance of maintaining an open, honest and objective policy on rival projects. Firstly, because it encourages a certain value for the user community, and secondly, because it fosters the development of coopetition strategies with aligned projects.

2. The user community

As explained in the first section of this module, the role of the free software user community is very important in the paradigm of free software development.

Both the users and the developers who form part of the community collaborate on the maintenance, support and evolution of the application over time, thereby ensuring the cohesion and stability of the project.

Consequently, their participation is essential for securing the project aims and should be considered as such by any money-making organisation seeking to exploit a business opportunity based on the production of free software.

In this sense, the relationship between community and business should be founded on the credibility and transparency of all actions and decisions taken, so that both parties can benefit from the relationship. Not surprisingly, the company's positioning with respect to products based on free software must be well defined and structured to encourage the creation of a community of collaborators around it.

Note that the user community is a dynamic organisation that evolves over time, so it will be necessary to set up management methodologies in order to maintain an optimal relationship. This management includes the establishment of procedures to identify the current status of the community, to assess the quality of contributions to the project by members, and to define legal aspects affecting these contributions.

The following sections will study each of these aspects in turn.

2.1. Community management

To secure the aims of the project, a company that undertakes a free software development project must organise its relationship with the user community carefully.

In the first section of this module, we looked at the main aspects underpinning a free software project. If a company acts as project founder, it will need to establish and organise a strategy to suit the business aims, though bearing in mind that it has to compensate for the collaboration it hopes to obtain from the user community.

Hence, as with any other free software project, issues such as credibility and transparency, among others, have a very important role in creating a community of users around the project.

Ben Collins-Sussman and Brian Fitzpatrick have identified and classified the different *Open Source* strategies that can be adopted by a company based on free software development at the OSCON 2007 conference entitled "What's in it for me?".

Recommended website

B. Collins-Sussman; B. Fitzpatrick (2007). "What's in it for me?"
(http://www.youtube.com/watch?v=ZtYJoatnHb8).

This classification characterises the two main components of the relationship between company and community:

- On the one hand, the orientation, structure and general operation of the project, and the company's responsibility in this.

- And on the other, the benefits and drawbacks for the company and the user community resulting from the selection of a specific strategy to implement the project.

Hence, the work of Collins-Sussman and Fitzpatrick is a guide to best practices in formalising a healthy relationship between the company and the user community.

In the following sections, we will briefly introduce the main features of this *Open Source* strategy classification.

Fake Open Source

This strategy is based on opening or releasing the application's source code under a licence not approved by OSI.

Recommended website

Open Source Initiative (http://www.opensource.org/).

It is not really an *Open Source* strategy because not only are thebenefits[4] lost, but some members of the community may even boycott the project.

(4)For example, software enhancement, project credibility or good relations between companies and users.

Nonetheless, the project can obtain media coverage and attract attention with a relatively low effort and cost.

Throw code over the wall

This is a similar strategy to the one above except that this time the company opens or releases the code under an OSI-approved licence, although it is still not concerned or does not accept responsibility for the future of the project.

In other words, by opening up the code and forgetting about it, the company portrays an image of poor credibility, since it releases an application for which there is no user community to keep the project alive. In this case, alternative communities may spring up to develop the software outside the business goals.

Develop internally, post externally

This strategy is based on developing the application internally within the company and publishing the progress in a public repository.

This time, the company improves both its public relations with the user community and its credibility in the world of free software. For its part, the community could collaborate on the project from time to time. Nonetheless, a totally internal development will encourage the development of parallel communities that do not follow the business calendar (which generates an element of distrust).

Open monarchy

This strategy is based on making public both discussions and the repository of the application, although the users with the rights to it are from the company.

In this case, the credibility and transparency of the companies and the input from the community are improved (which results in better code) but the company still has the final say on all decisions made. This constitutes a risk to the long-term maintenance of the community, including the possibility of a fork in the project.

Consensus-based development

This strategy exploits almost all possible relations between company and community, given that virtually everything is done in public.

In this case, the project is based both on distributed and decentralised decision-making and on meritocratic work systems among collaborators.

These features produce a model with high quality volunteers that is sustainable in the long run, since the company gains in credibility, transparency and reliability in the eyes of the community and other free-software companies.

Nonetheless, the short-term benefits are limited and the workload is significant. In this case, the role of the project leaders is relevant for the strategic operation of the entire organisation.

2.2. Community features

The community of free software users is a dynamic and evolutionary organisation in the sense that there are several factors that influence and shape its situation and future trends to varying degrees.

When considering a free software project, it is desirable to create an early and strong user community around the application, given that part of the success and aims of the project depend on it.

Once the community has been created, it is important to schedule activities that will not only keep it stable but also enlarge and evolve it, at least at the same pace as the product. Before we take any action in this regard, we need to ascertain the current status of the user community and its recent evolutionary trend in relation to the project.

Accurately identifying the current status of a user community can be relatively complicated in practice, mainly due to its qualities of distribution and decentralisation[5].

(5)This problem can be conveyed and illustrated with the problem of assessing the situation of a distributed or decentralised system at a given point in time (snapshot).

Nonetheless, we can take into account a series of indicators that will allow us to establish a sufficiently realistic approach for making decisions on this subject.

The article "Assessing the Health of a FLOSS Community," by Crowston and Howison, describes a simple but effective guide to identifying and assessing the status of a community of free software users. This guide considers the main indicators that should be taken into account when assessing the health of the community and, by extension, the free software project.

Recommended website

K. Crowston; J. Howison (2006). "Assessing the health of a FLOSS Community" (http://floss.syr.edu/publications/Crowston2006 Assessing_the_health_of_open_source_communities.pdf)

The following sections will briefly introduce some of their findings.

Life cycle and motivations

Diverse authors concur that projects are initiated by a small group of founders before being structured and publicly developed.

Once the project has been launched, a second phase should begin allowing for the progressive refinement of the initial concept. In other words, the sharing of ideas, suggestions and knowledge must revolutionise the original concept. This process cannot be completed without the cooperation of the free software community.

Moreover, the participation of members of the community in the project is chiefly motivated by intellectual development, the sharing of knowledge, interest in the application, the ideology or philosophy behind the project or free software, reputation and community obligation.

Structure and size of the community

The user community of an application based on free software can be structured in many ways, taking into account the actions and decisions of the project founders and the features of the application and/or its production.

In general, we can consider an application's user community to be healthy if it has a functional hierarchical structure aligned with its aims around an active core of developers.

Hierarchical structure

This concept can be compared to the structure of the layers of an onion (onion-shaped), whereby the most active members of the project are at the core and the less participatory members are found in the outermost layer.

Broadly speaking, we can identify the following types of member in a project:

- Developers of the application kernel, with write permissions on the repository and a significant history of contributions to the project.
- Leaders of the project, who motivate and lead the project and its user community to maturity and stability.
- Developers in general, who contribute code but have no write permissions on the repository. They often perform review tasks.
- Active users, who test the application, report bugs, draft documentation and link the project up with passive users, among other activities.

Note

This initial classification of typologies is not a closed structure, since each project will adapt it to suit its particular features.

Development processes

The process of free software development can often be inadequately formalised in projects, mainly due to the absence of roadmaps, explicit work assignment or the lack of prioritisation in the application's features.

The organisation of the project is relevant to the functioning and coordination of production, although a certain degree of duplication of effort could be considered a positive sign of the relationship and involvement of the community with the project.

Likewise, the cycle of evaluation and subsequent acceptance of contributions from community members to the project provides accurate information on its health. For example, the rejection of a contribution can reveal a cohesive and qualitative vision of the project in the long run.

2.3. Quality management

The quality of free software has sometimes sparked debate between its advocates and detractors, particularly concerning aspects such as the openness of the development model or the skills level of collaborators who contribute to the project, for example.

As with any software project, free software production should establish measures for quality control throughout its life cycle. In other words, we must be able to assess its quality and compare it with the levels expected at any stage of production or exploitation and from any angle (founders, users or community).

While the openness and decentralisation of the model of free software development allows for quality control and management mechanisms, they are not a solution in themselves and planning should not be overlooked because of these features.

To develop the quality aspects of free software production, we will study the Dhruv Mohindra's article "Managing Quality in Open Source Software", which conducts a detailed analysis of quality control in free software environments. In the subsequent sections, we will review the main ideas of the article.

Recommended website

D. Mohindra (2008). "Managing Quality in Open Source Software" (http://www1.webng.com/dhruv/material/managing_quality_in_oo.pdf).

Quality in free software

In general, the quality of a software solution can be assessed both by its architecture or internal design and by the functionality it provides to the user.

The specific features of openness and decentralisation of the free software development model create an infrastructure that allows for quality management policies to be established through the identification and resolution of problems, among other aspects. Still, a lack of clarity and/or structure in production processes can sometimes generate unexpected results.

Assessing quality

There are several formal methodologies and metrics for assessing the functional quality of an application. Quantifiable metrics depend largely on the typology of the software itself, so they must be chosen in accordance with the features and aims of the application.

The free software community plays an important role in non-quantifiable quality: firstly, in the tests performed by the quality team, and secondly, in the activity of the users of the application, who report evidence of malfunctions or for product enhancement.

In this sense, the decentralised and distributed nature and operation of the user community is important for increasing the quality guarantees of the production process.

Control and review

An important factor in end product quality is the control and review of the entire development process. In general, free software production projects use version control systems to efficiently and effectively support the evolution of the diverse project components.

There are different ways to organise the control and review of the evolution of the software and its branches of development and repositories, among other aspects. In all events, though, it is a good idea to adapt the production methodology and systems for the control and review of changes to the specific features of the project and the product being created.

Free software myths

Despite the passing of time, there are still some myths, both positive and negative, associated with free software that can influence its assessment to different degrees.

These myths have no solid foundation on which to base a coherent and sustainable quality management, so we need to identify and evaluate each one individually.

We will then discuss some common myths associated with the quality of free software.

- The fact that the source code is public does not guarantee that it is secure and/or of good quality, as this depends on the community interest and reviews.

- Feature freezing does not increase the stability of the application in itself, because the important thing is to write good code from the start.

- The best way to understand a project is not to correct its possible shortcomings, as the documentation is significantly better for this purpose.

- Generally, users do not have the latest version of the repository with updated bug-fixing.

Broadly speaking, the testing and review processes, and the public discussions and hacker culture specific to the user community must be complemented by the active planning and management of production quality.

This management should seek to fill any gaps in one or more aspects of the product, e.g. production planning, development of the features or the documentation of the application.

Additional quality considerations

In general, both the release of the source code and the incorporation of error handling systems and the sharing of responsibility for the product among all those involved are key aspects of quality management.

Hence, it is also important for the overall quality of the project to consider transparency in all actions, trust the development team, review and test all parts of the source code and promote both the peer-to-peer philosophy and the importance of doing things well from the start.

2.4. Legality and contributions

In a project based on free software with participation from the user community, the legal management of the contributions of each member involved is particularly important.

This management is crucial both for the project founders and for the members of the community, as it establishes the features of the authorship and ownership of the rights to the resulting code. Its relevance is also strongly influenced by the implications that the combination of codes from different authors could have on a single product.

To develop these concepts, we will refer to section 2.4 "Authors and holders of rights" of the teaching materials for the subject *Legal aspects and the features of exploitation of free software*.

Recommended website

M. Bain et al.(2007). *Aspectos legales y de explotación del software libre*. Universitat Oberta de Catalunya (http://ocw.uoc.edu/informatica-tecnologia-i-multimedia/aspectes-legals-i-dexplotacio/materials/).

The author

The author of a work is the natural or legal entity that creates the work, so authorship of the original creation is irrevocably assigned to this person.

With works by several individuals, there are a number of possible situations:

- A collaborative work is the unit result of a composition of different parts that can be exploited independently.
- A collective work is the collection of diverse contributions that cannot be exploited independently.
- With a commissioned work (or one with financial compensation), authorship lies with the person or entity that carries out the commission.

In free software, authorship depends largely on the above considerations, taking into account that the transfer of ownership can sometimes be useful and practical.

Moreover, the conditions under which derivative works are created (pre-existing content) may vary materially because of both the author and the work itself. In all events, free licences must specify the conditions of the derivation and redistribution of the works.

The original owner and the derivative owner

The original owner of the work is always the author. However, some rights over the work may be transferred to other individuals or entities.

In this case, the recipient of the transfer of part of the rights to a work becomes the derivative holder thereof. Note that only the holder of a particular right may licence that right.

Identifying the holder

In order to exercise the above rights, we must be able to identify the author of each work. This can be difficult in free software because the contributors to the project may be many and varied.

To solve these problems, projects based on free software keep lists of the authors who have contributed to them. Sometimes, these projects may require the transfer of all or part of the rights before the contribution can be accepted.

3. Case study

In the previous sections, we have examined both free software projects and the management of user communities. Both sections describe the key aspects of free software production from the point of view of project management.

To complete the module, this section will go into further detail on many of the ideas and proposals described above before moving on to study a specific case of a company based on free software.

The following sections are intended to serve as a guide for identifying and clarifying how a company based on free software production implements its particular methodology, formalises and manages its relationship with the user community and addresses the many decisions that need to be taken as time goes on.

In this section, we will study the case of Openbravo, S.L.

3.1. The company

Openbravo, S.L. is a company that develops professional solutions based on free software for business.

Note

The information in this section has been taken mainly from the corporate website (http://www.openbravo.com/).

Business model

The business model exploited by the company is that of providing services for the products it develops. As we explained in other modules, its business strategy is based on associationism and coopetition between companies in order to exploit the same business opportunity.

Business strategy

The business model is implemented by partners that provide services to end customers (such as customisation and support). In a sense, this particular hierarchy between producer, distributor (or partner) and client establishes an atmosphere of cooperativism with common goals.

To complete the strategy, the company publishes a manifesto as a sort of statement of intent, which combines aspects of free software (for example, transparency, openness and collaboration) with the company's third-party commitments (such as free access or contribution management).

Management and leadership The company's management combines tasks that are internal and external to the organisation, both in its management team and its Board of Directors, which is the result of foreign investment injected into the company combined with its particular methodology based on free software.

3.2. Products

Openbravo produces two free software solutions that can work independently of each other or in combination. Both products are distributed under free licences and can be downloaded directly from the Internet.

The products offered by Openbravo are:

- **Openbravo ERP**
  Openbravo ERP is an enterprise resource planning system in a web environment that integrates various management functions, such as supply, warehousing, production and accounting, in a modular way.
  The product is licensed under MPL 1.1 and can operate in different environments and database systems and be integrated with Openbravo POS.

  One of the highlights of the vast amount of information provided on the product is the roadmap of the project development.

Recommended website

Mozilla Public License 1.1 (http://www.mozilla.org/MPL/MPL-1.1.html).

Main features of Openbravo ERP

http://sourceforge.net/projects/openbravo/.

- **Openbravo POS**
  Openbravo POS is a point-of-sale terminal system that can be integrated with Openbravo ERP.
  The product is licensed under the GNU/GPL licence and can run in different environments and with different database systems. It was especially designed for touch-screen terminals.

  The available product information includes the roadmap of the project development.

Recommended website

GNU General Public License (http://www.gnu.org/licenses/gpl.html).

Main features of Openbravo POS

http://sourceforge.net/projects/openbravopos/.

3.3. The user community

The community of free software users plays an important role in Openbravo's business strategy. The following sections examine its main aspects.

Open Source Strategy

To identify Openbravo's Open Source strategy, we need to consider the specific features of the product development methodology and the business structure used to exploit them.

The kernel of both products is primarily developed internally within the company and public repositories and an active user community are maintained around it. For the development of complements, customisations and extensions to the original product, both the user community and partners play a part.

Partners must be examined separately because they correspond to the exploitation of an opportunity by a different organisation.

Nonetheless, Openbravo uses an Open Source strategy that combines different orientations:

- In its product development and review, the strategy used is similar to *Open Monarchy*, mainly due to the internal development of the product kernel, the public repositories of source code, the company's final acceptance of changes to the kernel and the planning of product development (for example, the established roadmaps).

- In the development of complements (documentation, etc.), the strategy is more similar to *Consensus-based Development*, due mainly to its development within the user community.

- And lastly, the strategy for the development of extensions and customisations depends on the developer who implements them. If they are projects carried out within the community (using the resources offered by Openbravo), they are possibly more similar to the *Consensus-based development* model, while if they are developed by partners, they will depend both on the strategy in question and the features of the development.

Partner strategy

If the partner develops extensions of the original product, the *Open Source* strategy will depend as much on its business philosophy as on the features of the product (for example, the MPL is more flexible with proprietary modules than the GPL).

Community structure

The Openbravo ERP user community is defined and structured as a meritocratic system: there are several levels of collaboration and each is defined by the knowledge required for this level, the amount of contributions made, responsibilities and privileges.

For Openbravo ERP, there are three different collaboration profiles (developers, functional experts and testers), while in the case of Openbravo POSITION, there are only developers. The members of the user community organise themselves and are distributed into active projects in the community.

Resources available to the community

Openbravo offers a range of resources (some in more than one language) for the community and for its partners or general users. These include:

Recommended website

All of the resources mentioned can be accessed from the company website (http://www.openbravo.com/).

- Corporate website
- Partners area
- wiki project
- Portal for the Openbravo user community
- Employee blogs
- Forge for products (Openbravo ERP and Openbravo POS)
- Bug tracker
- University
- Mailing lists
- Openbravo code repository
- Openbravo news service

The user community can refer generally to a specific guide in the wiki explaining how to collaborate with the project. It also has an exhaustive list of communication channels that it can access. The roadmaps of each product complete the resource guide for the user community.

3.4. Positioning and evolution

The company was founded as Tecnicia in 2001. In 2006, it obtained funding in excess of six million dollars, when it was renamed Openbravo. That same year, it released the source code of the products it develops under free licenses.

In May 2008, the funding round amounted to over twelve million dollars, with investors including Sodena, GIMV, Adara and Amadeus Capital Partners.

Over the years, Openbravo has won several business and free software awards and received grants from the Spanish Ministry of Industry, Tourism and Trade's PROFIT programme to promote technical research.

Both the company and the user community display a positive trend in development, given that the project is currently one of the twenty-five most active on SourceForge with more than one million cumulative downloads in early 2009.

Recommended website

For the most active projects on SourceForge:
http://sourceforge.net/top/mostactive.php?type=week.

Summary

In this module, we have described the main features of the creation, management and maintenance of free software development projects, taking into account the participation of the user community.

In a sense, the foundations of free software production do not differ that much from the methodologies of traditional software development. However, the features of open code and the presence of the user community shape its operation, making it unique in many respects.

With regard to the project per se, we must stress the importance of identifying, defining and structuring both the functional aspects of the project (infrastructure, version management and coordination measures) and those concerning free software (credibility, transparency or typologies of participation).

In addition to these aspects, there are factors associated with the free software community, such as the company's strategy for community management (Open Source strategy), the product life cycle and methodology, quality management and the legal aspects of user contributions.

Lastly, we described a case study that is representative of many of the aspects we have seen in the different sections.

Bibliography

Bain, M. et al.(2007). *Aspectes legals i d'explotació del programari lliure*. Universitat Oberta de Catalunya <http://ocw.uoc.edu/informatica-tecnologia-i-multimedia/aspectes-legals-i-dexplotacio/Course_listing> [Consulted in February 2009].

Collins-Sussman, B.; Fitzpatrick B.(2007). *What's in it for me? How your company can benefit from open sourcing code*. OSCON: 27 July 2007 <http://www.youtube.com/watch?v=ZtYJoatnHb8> and slides <http://www.red-bean.com/fitz/presentations/2007-07-27-OSCON-whats-in-it-for-me.pdf> [Consulted in February 2009].

Crowston, K.; Howison, J.(May 2006). *Assessing the Health of a FLOSS Community*. IT Systems Perspectives (pp. 113-115). <http://floss.syr.edu/publications/Crowston2006Assessing_the_health_of_open_source_communities.pdf> [Consulted in February 2009].

Fogel, K.(2005). *Producing Open Source Software: How to Run a Successful Free Software Project.* <http://producingoss.com> [Consulted in February 2009].

Mako, B. (2001). *Free Software Project Management HOW TO*. <http://mako.cc/projects/howto> [Consulted in February 2009].

Mohindra, D. (2008). *Managing Quality in Open Source Software*. <http://www1.webng.com/dhruv/material/managing_quality_in_oo.pdf> [Consulted in February 2009].

Openbravo <http://www.openbravo.com/> [Consulted in February 2009].

Raymond, E. (1997). *The cathedral and the bazaar* <http://www.catb.org/~esr/writings/cathedral-bazaar/> [Consulted in February 2009].

Tawileh, A. et al.(August 2006). *Managing Quality in the Free and Open Source Software Community* (pp. 4-6). Proceedings of the Twelfth Americas Conference on Information Systems. Mexico: Acapulco. <http://www.tawileh.net/anas//files/downloads/papers/FOSS-QA.pdf?download> [Consulted in February 2009].