Management of Interorganizational Open Source Software Projects - Lessons from a Case Study in Developing Country Context

Jussi Nissila
Jussi Puhakainen
Turku School of Economics
Pasi Malinen
Turku School of Economics

Follow this and additional works at: http://aisel.aisnet.org/amcis2007

Recommended Citation
http://aisel.aisnet.org/amcis2007/269

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2007 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
Management of Interorganizational Open Source Software Projects – Lessons from a Case Study in Developing Country Context

Jussi Nissilä, University of Turku, Finland, jussi.nissila@utu.fi
Jussi Puhakainen, Turku School of Economics, Finland, jussi.puhakainen@tse.fi
Pasi Malinen, Turku School of Economics, Finland, pasi.malinen@tse.fi

Abstract

This paper focuses on the management of interorganizational open source software projects. It presents a case of an ongoing United Nations-funded project, aimed at developing an information system for the management of agricultural co-operatives in developing countries. Drawing on the results of this case study, the paper presents normative guidelines for any researcher or practitioner taking part to the project management of publicly funded development aid OSS projects. Key findings stress the need of establishing a viable business ecosystem, which contains multiple actors. Also the importance of open source specific business models is pointed out.

Keywords: Open source software, business models, software project management, case study
Introduction

Open Source Software (OSS) is a tempting choice for many software projects in developing countries. The lack of expensive licenses for software and independence from major software producers attract projects with limited resources and even countries, who wish for political or economical reasons base their ICT-strategies on “free” software (see for example Weerawarana and Weeratunge 2003, Noronha 2002, Wright 2006). Likewise as naturally the costs of software project do not revolve only around software licenses, nor is the cost the only issue in the growing popularity of OSS in developing countries. “I don’t think the cost (alone) is an issue…But more than price, what matters is the application development. The idea of the openness should be kept there. Openness and sharing…these are great values in themselves”, Pierre Dandjinou, ICT-D Policy advisor for Africa (quoted by Noronha 2002).

In this paper we will present a case of publicly-funded open source project. Food and Agriculture Organisation of the United Nation (FAO) with the financial support of the Government of Finland is undertaking an inter-regional project titled "Development of a Member and Management Information System (MMIS)". The FAO project identification number is GCP/INT/986/FIN. The objective of the project is to enhance the capacities of agricultural co-operatives and producer organizations in developing countries to improve their business efficiency and competitiveness in national, regional and global markets through the development and dissemination of a low-cost and marketable prototype computerized management and member information system. The development of this open source licensed prototype software builds on an existing system recently developed within a FAO Technical Cooperation Programme project in one pilot dairy co-operative in Kenya. The MMIS has recently been given the name CoopWorks – Business Management Information System for Producer Organizations. Besides Kenya as a leading country, this inter-regional project will also work with pilot producer organizations in one further African country (Tanzania, Zambia or Uganda) and in Vietnam.

Open source software is traditionally developed “to scratch an itch” of the developers themselves (Raymond 2001). This means that software is developed, mostly by voluntary individuals, to solve a problem the developers have themselves. Increasingly, open source software projects are founded also based on commercial motivations, where an entrepreneur or enterprise strives for profit by providing services or complementary products together with open source software. However, there are many cases where another party is scratching the itch of another, but where also economic sustainable development of the project and high impact of the software are key issues to be taken care of. Unlike voluntary or commercial OSS projects mentioned before, these kinds of projects are usually interorganizational; i.e. there are many developers from different organizations involved, and a complex stakeholder network. This article takes a look at the management and leadership issues of interorganizational open source software projects. Although there is relatively large amount of literature on OSS project management (see for example Senyard and Michlmayer 2004, Michlmayer et al. 2005 for typical examples) there are relatively few articles focusing on publicly-funded interorganizational projects. We take a look at the management and business issues of open source software projects, and present a case study in order to illustrate some key issues in the management of funded inter-organizational open source projects. The case study is based in developing country context, which sets the practical realities of the case, but the lessons may be useful in the management of similar OSS projects in any context.

Background

The seeds of this research project lie in the late 80’s research project, in which the researchers of the Turku School of Economics, Finland, surveyed Kenyan agricultural coffee and dairy sector co-operatives. The project was funded by Food and Agriculture Organisation of the United Nation (FAO). The general aim of the project was to give normative advice how Kenyan farmer owned co-operatives could enhance their capital formation and management efficiency. There is a long tradition in Scandinavian research on third world co-operatives. For example the Scandinavian Institutes of African Studies published a book ar early as 1972 focusing solely on co-operatives and efficiency (Widstrånd 1972). The final report of the project suggested computerization as one of the key elements in accomplishing this aim (Järmsén et al. 1999). A decade later this recommendation was put into action.

Current research is part of a two-year project, which aims at enhancing the capacities of agricultural co-operatives and producer organizations in developing countries to improve their business efficiency and competitiveness in national, regional and global markets through the development and dissemination of a low-cost and marketable prototype computerized
management and member information system (MMIS) for global use and further development. Normative guidelines will be prepared to guide stakeholders in member countries for further dissemination of this universal prototype system. The project is expected to contribute significantly to strengthening FAO’s Rural Development Division’s new agricultural co-operative computerisation programme and strengthening of rural producer organizations. The development of the prototype system is build on existing systems and experience from co-operatives supported by FAO as well as other organizations and their projects in Africa and Asia and on case studies conducted in three selected pilot agricultural co-operatives in Africa and Asia.

A preliminary scoping study of the existing level of co-operative computerization in the three pilot countries is carried out first. This study, together with information gathered on the computerized system currently being developed in Kenya (TCP/KEN/2907) as well as experiences in other countries that have already some experience in co-operative computerization (e.g. India, Thailand, Korea, and Argentina), help to determine the product development strategy. Based on the strategy Turku School of Economics, TSE), working in close consultation with national IT consultants, assist in designing and evaluate the prototype-computerized system for management and member information. The system design, foreseen to be modular, caters for various types of co-operative activities and will be tested at various sites with the help of local IT experts to make modifications as appropriate to further improve functionality. The first step will be the design and testing of a general business accounting module suitable for co-operatives. Subsequently, add-on modules will be developed and tested to cater for the various activity-specific features of different co-operative businesses. The software is open source licensed software (OSS), and therefore, can be developed further to meet various needs of potential co-operatives in all over the world.

After a full cycle of testing and implementation by the end of year 2, the results of product development and field-testing: a beta version of a universal prototype system capable of handling the management and member information needs of a range of agricultural co-operative businesses, will be reviewed in national workshops that will contribute to the further refinement and operationalisation of this normative prototype system. At the outset, it seems appropriate to include all the various activity-specific features in the final universal prototype system. The user could then select from the programme menu of the prototype system, which particular features would need to be incorporated in the information system of a specific co-operative.

The system

CoopWorks software technologies

The CoopWorks 1.0 is a three tier web application, which is designed to run on server/client architecture. It has been developed with Microsoft ASP.net technology using VBScript (Visual Basic Script Edition) and HTML language. Platform requirements for the CoopWorks 1.0 are Microsoft Server, MS SQL 2000 database and Windows operating systems and Internet Explorer browsers in client PCs.

ASP.net is a set of web development technologies, which includes an event-driven GUI (graphical user interface), and it can be used to build dynamic web sites and web applications. ASP.net is the successor to Microsoft's Active Server Pages (ASP) technology. However, ASP.net encourages the usage of an event-driven GUI more than the conventional web-scripting environments, such as ASP and PHP.

When using the ASP.net web development technologies the code can be written by using any of the different programming languages supported by the Microsoft’s .NET Framework, including C#, VBScript, JScript and open-source languages such as Perl and Python. The CoopWorks 1.0 is written with VBScript. VBScript is not fully compatible with other browser than Internet Explorer, which is a browser that comes together with Windows operating systems.

Weaknesses of current technology choices

Currently the CoopWorks software architecture is based on proprietary software technologies, even if the CoopWorks software itself is licensed under OSS license. Dependence on a certain vendor and product, in this case Microsoft Server and Microsoft SQL database and also Internet Explorer, is prone to provide problem such as:
- High cost of licensing. Proprietary software licensing fees can be a significant part of the total cost of ownership of information systems, especially in the case of developing countries. The licensing fees tend to be higher in cases, where the customer does not have significant negotiation power.
- Lack of choices. Any choice of technology will limit the choices that can be made regarding the implementation of the information system. This can be a problem, especially regarding legacy systems, where a co-operative already has some computers in use. In this case they may be forced to abandon the old computer systems and reinvest in technology specific systems.
- Customer lock-in. Once a choice in the direction of vendor specific technology is made, the customer may find changing the technology and the vendor increasingly difficult, due to the effort and costs of changing the technology and all related processes and computer equipment.
- Forced changes. If the technology vendor decides to change the technology in question, the customer may have to make the same changes due to the lack of support and benefits related to the old version of the technology. These changes may have negative effects in terms of adaptation costs, increasing maintenance costs, and limited / unwanted functionalities.

In addition to the issues above, it has to be noted that high cost of the Microsoft platform may motivate software piracy. In addition to the obvious moral and political problems illegal use of software causes, it might also cause practical issues to co-operatives. For example, Microsoft has been trying to decrease piracy by making it difficult to update unlicensed copies of its products; leaving systems with pirated copies insecure and buggy. In addition, using pirated software leaves this kind of co-operatives prone to financial risks, if the copyright owners find out about the use and decides to prosecute the co-operative.

The decisions concerning the technologies of CoopWorks were made when there was no clear vision of the project future, and serve the functionalities of the information system well, but may cause problems in the future. The issues mentioned above influence the adoptability (regarding costs of licensing and information systems changes) and flexibility of the CoopWorks project. Committing to a certain technology vendor may also lead to diminished independence of the project.

### Possible technology changes

As discussed in the previous section, the current situation regarding technology choices causes several problems, which might have critical consequences for the future development of the project. In addition to the problems the co-operatives are subject to face, technology dependence may cause the project to fork into two or more separate and difficult to manage projects using different technologies, if the user/developer communities in other countries decide to choose another technology. Researcher did suggest that serious thought is given to the possibility of developing the software towards platform independence, meaning that CoopWorks could work on different operating systems, web servers, SQL servers and also with different clients. This study is now underway.

With platform independence the adoptability of CoopWorks would increase, if potential users could use their already existing information systems infrastructure as a platform for CoopWorks, or they could invest on a platform most suitable for them. This would also mean lower costs for them. Also, the envisioned developer community would involve more people globally, since the development work could be done with a wider range of technologies. In addition, if CoopWorks could be used on an open source software platform, it would benefit from the advantages provided by the open source software model.

The current version of CoopWorks is licensed with an open source compatible software licence, and is therefore also open source software. However, since CoopWorks is relying on proprietary software products, the project gains only partly the benefits of open source software licensing and community development. The role of open source software licensing and the benefits it provides could be discussed more, and information about open source software could also be provided in the CoopWorks website. However, the first issue would be to study the possibilities and feasibility of developing a platform independent and/or open source compatible version of CoopWorks. Then CoopWorks could properly be called and marketed as an open source software system.

The use of CoopWorks requires purchasing proprietary software, because the platform requirements for the CoopWorks 1.0 are Microsoft Server with an MS SQL 2000 database. The CoopWorks 1.0 is not compatible with Linux operating system or an open source software database applications. Even the client PCs, which are connected to server, should run on a Windows operating system. In order to use CoopWorks in open source software environment - for example with a Linux operating system, MySQL or PostgreSQL database and Mozilla browser - some modifications should be done. Instead of writing different versions for different databases and operating systems, it is possible to develop a platform independent version of CoopWorks by choosing different technologies and programming languages.
One possibility is to re-program the CoopWorks 1.0 with PHP, which is an open source programming language and an alternative to Microsoft’s ASP.net. Another solution is to use ASP.net but choose another scripting language than VBScript. The SQL queries for the database should also be checked to make them compatible with other databases, including open source software databases such as MySQL or PostgreSQL. This is possible, for example, with Open Database Connectivity (ODBC), which provides a standard software application programming interface method for using database management systems and is designed to be independent of programming language, database system and operating system.

Costs, feasibility and the best way of the development of a platform independent and open source software compatible version of CoopWorks should be studied carefully. The most significant issue is to design the software on a high level of abstraction. A careful high level design is essential in order to achieve platform independence, and it is also recommendable to better support the creation of other modules. It is possible that full platform independence is not an optimal solution, due to for example the structure of SQL queries etc. However, qualities such as high modularity and easy of portability should be key aspects in the software design. Investments in the high level design can also make it easier to develop and integrate other modules developed from the CoopWorks base code.

Research approach

This article presents a case study on the management of interorganizational open source software development projects in a developing country context.

The authors recognized this as a unique opportunity to study this complex project in a real world context. There is limited amount of previous research on interorganizational OSS development projects, their business or management issues, and also quite little research on ICT development in developing countries. The authors couldn’t recognize a single theoretical framework that could explain this kind of complex case. Therefore, we used exploratory case study method to study the phenomenon. According to Yin (2003, 15-30) exploratory case study is suitable to explore situations in which the intervention being evaluated has no clear, single set of outcomes, and the existing knowledge base about the subject may be poor.

Qualitative methods were used in this case study. Case studies draw from different sources of evidence. According to Yin (1990, 85-95) case studies uses six sources of evidence, namely 1) documents, 2) archive material, 3) interviews, 4) direct observation, 5) participant observation, and 6) physical material. This case study the authors used a multitude of source, including related project documents and reports, interview of project employees and project stakeholders, direct and participatory observation of the project management, project context and the project software in order to have a complete picture of the project. In addition, a large secondary data was used and desk research was conducted. All this aimed at triangulation (Yin 1990) of different kind of data and increasing the validity and reliability of the study. Actual research and data sources will be presented in more detail in the actual case part of this paper.

General aims for the case-research in Kenya

The research project has four general main aims, each of which had separate research questions. Due to the nature of this paper, we will focus on these general aims.

The first, and in our minds, the most important one was to collect information and form an understanding of the viability of the GCP/INT/986/FIN on a commercial basis in the long run. That is, what role and functions are required from the local business ecosystem; what is the status of the current open source community and what potential does it have to support CoopWorks; what other agents are there that could help to support and drive the project.

Second, to collect information and form an understanding of the potential problems and advantages that different choices made regarding software CoopWorks have on the project and the project clients. That is, what are the potential effects for including or excluding certain functionalities, elements, and technologies on the to be expected project results and its stakeholders; how do these elements affect the commercial viability; how do these elements affect the maintainability of Coop Works and how does this affect the expected project objectives. Challenges in adopting and using CoopWorks will be analyzed.
It was also necessary to assist FAO in identifying training needs to support the uptake of CoopWorks in Kenya and other countries and assist in developing a draft capacity building plan for possibly training of trainer types of capacity building activities.

For the co-operatives, we aimed in developing and test a simple baseline survey instrument to enable co-operatives to assess how the introduction of CoopWorks is influencing socioeconomic development of co-operative members. TSEBA mission was to develop the draft instrument, including the research questionnaire, method and process and undertake a quick test of this during the inception mission in one co-operative (Tulaga Farmers Co-operative Society) and prepare a final version that could be available through www.coopworks.org and that could be used by co-operatives and rural producer organizations themselves to monitor the socioeconomic impact of introducing CoopWorks.

Kenya: three missions

The first research mission to Kenya took place in December 2005. The general aim of the research team was to get acquainted with the previous projects and to do a quick evaluation of the first version of the prototype system. Actual research included interviews with relevant stakeholders (ministry for co-operatives, local FAO, Kenyan ICT-federation) as well as observations and interviews in the actual pilot co-operative (Tulaga dairy co-operative). The system was evaluated according to the classical DeLone & McLean-guidelines (see DeLone and McLean 1992, 2002, 2003). The results of this rapid assessment were promising. The pilot co-operative was very pleased with the system, claiming marked organizational impact on their accountancy and member management operations. For the first time, the management was able to see in almost real time the amounts individual members were producing. It is noteworthy that a typical agricultural co-operative has around 200-500 members, who each produce relatively minor amounts (a family with one cow might produce daily 2 kg milk). The amount of transactions is thus staggering compared to the value of transactions. The information system gave both longer range planning possibilities as well as increased trust towards the operations of the co-operative, since members were able to know constantly how much the co-operative owned them. The board of directors claimed that the membership of the co-operative had tripled after the launch of the system.

The second mission took part in May 2006 and included:

A desk research which included a literature review and statistics available in the Internet was conducted. The objective of the literature review was to combine the theories and ideas from the fields of business management, development studies and from field of information system science, especially from the literature about open source software. Research and studies which combine open source software and developing countries are still rare. Thus, the objective of the literature review was to introduce these two fields and provide suggestions about the potential benefits open source software can provide for developing countries. The emphasis is on the potential benefits regarding socioeconomic development and on the role of open source software principles.

Secondly, TSE collected empirical material with theme interviews and open discussions, and by participating on the conference Coop Management Turn-around through ICT -seminar in Nairobi on 24 May 2006. Theme interviews and discussions took place between 22 and 26 May 2006 in Nairobi and on 25 May in Tulaga farmers co-operative society. In addition, the documentation of the member and management information system project was used.

During the inception mission TSE interviewed key informants and other stakeholders in formal and informal settings. These included:

- Users of CoopWorks: Tulaga Co-operative Farmers Society
- People involved in the design and implementation of the project and the software: WebDiscount Technologies, and Tulaga Co-operative Farmers Society
- Other stakeholders which have a role in marketing, dissemination, and planning the future of the CoopWorks project: Food and Agriculture Organisation (FAO) and its contractors in Kenya, Kenya national federation of agricultural producers (KENFAP), Ministry of Co-operative Development and Marketing (MoCDM)
- Kenyan ICT sector and open source software communities: The Free Software and Open Source Foundation for Africa (FOSSFA), WebDiscount Technologies, Kenyan ICT federation (KIF)
- Other stakeholders: Embassy of Finland in Kenya, the ambassador; University of Nairobi.

Final mission took part in December 2006 and included interviews and discussions with the key stakeholders, focusing mainly in the continuation of funding and relevant issues.
Community issues

Business ecosystem

Currently in Kenya, many different governmental, NGO and business organizations are interested and involved in the project. This very positive, because no single organization can provide all the necessary services needed for a co-operative to start computerization to the full use of CoopWorks, ranging from basic PC skills training to implementing CoopWorks.

The first issues is awareness raising (marketing) of CoopWorks. This activity could take advantage of many different actors, including co-operative and agricultural apex organizations, which see clear benefits from CoopWorks to their members and the industry in whole). The ministry of Co-operative Development and Marketing has also a wide access and influence all over the country, which is useful. Private enterprises acting as service providers have pure commercial interests in marketing the software. And finally, word-of-mouth from a co-operative to co-operative, especially in fairs and conferences, may be very efficient, in a case of success stories.

Co-operatives new to computers need help and guidance in making business and investment plans for computerization, as well as in acquiring the basic infrastructure (such as electricity, buildings) for it. This type of services should be provided by a neutral party, such as an apex organization, which does not have a direct commercial interest in selling the infrastructure to the co-operative. TSE sees that these activities could join forces with other programmes (in Kenya; by the UN, NGOs or other organizations) supporting the infrastructure building and computerization of countryside and co-operatives.

Basic PC skill training may be taught by several parties. There are plenty of semi-commercial colleges in Kenya providing these kinds of services either at their premises or at-site. It is also possible that computerization programmes could offer training, or these services may be offered by the same ICT companies involved in providing the other services.

It is probable that offering the services directly related to CoopWorks are best provided by ICT companies of Kenya. This is because no single organization can offer these services in all the country, with flexibility and sustainability, which is only possible when there are apparent commercial opportunities. Therefore, the commercial viability of marketing and providing services to CoopWorks should be guaranteed. Also, the adoption of the CoopWorks business should be actively encouraged by the project and other key actors, for example by providing introduction, training and certification services to potential CoopWorks service providers.

The companies providing ICT services such as training or software development projects are mostly located in big cities. Thus, investments in ICT are also focused towards urban centers whereas 80% of the population is in rural areas. Encouraging the private sector to broaden markets requires government support to make it economically viable to invest in small and dispersed markets (Kenya ICT policy 2003). The location of ICT service providers is a problem for agricultural co-operatives which are located in rural areas. Depending on the distance and the quality of the roads to the nearest city, for example staff training costs become much higher because trainers need to travel many hours to come to train the stuff of the co-operative.

There are only few companies, which can provide open source software services is Kenya at the moment. Thus, the support for open source software products running on Linux operating system is very limited. However, the situation might be changing. There are many reasons why students, individuals, and companies are interested in open source software, and there are also Linux user groups forming in universities and elsewhere (see chapter 5.5 for detailed discussion). The situation of open source software skills in Kenya is therefore improving, although it would benefit of support from the government and international organizations.

Support for commercial activities by the government and by the UN can be considered very beneficial to Kenya in general, since the country suffers from loss of highly educated people due to lack of opportunities. Also, supporting entrepreneurship offers citizens direct means to improve their living conditions and other people at the same time. The project can support the development of co-operatives, computer infrastructure, computer and investment skill, as well as understanding of the business opportunities present by providing guidelines on the CoopWorks website.
Business models

Open source as viable business

There are many kind of OSS; complete Linux operating system, office applications, web servers etc. Open source development is also somewhat different from traditional software development. Open source licensing allows growth of development communities, where software is developed together by individual programmers, software companies, and other organizations working together voluntarily. The aim is to create better software, which can be used by all parties.

Open source as business was then discussed with the stakeholders. Main differences of open source business with traditional software business are that there are no licensing fees and no customer lock-in. These are good news for the client, and also for the OSS entrepreneur, because in this manner clients are more willing to adopt the software; a bigger customer base means more business opportunities; free software provides a basis of business activities.

There are in fact many open source business models, which are based on the fact that users have many needs in addition to the software. Therefore there are also many ways to make business:

- Selling services to facilitate OSS use
- Selling hardware to be used with OSS
- Selling proprietary (not-OSS) software to be used together with OSS
- Contracted software development

These may be provided for either:

- Already existing OSS
- Own software licensed as OSS

There is also business in providing consultation, implementation, customization, training and support for the software, as well as total solution packages (computers, other hardware, services). Services are in fact a big part of the total cost of acquiring and using an information system.

Finally, there was discussion about the rationale for participating to a developer community. These included adding value to the software product, which brings benefits because more value for the customer means more potential revenue, and the better the product – the bigger the market. Participation also develops in-house knowledge, leading to competitive edge when providing services.

Open source revenue models

There are indeed many different ways of doing business with open source. Whether an enterprise involved in the open source business chooses to license its own software product as open source, or tries to benefit from existing OSS products, the ways of doing money with open source are basically the same. These ways include selling services to facilitate OSS use, selling connected hardware, and selling commercial closed applications to use with OSS. Different revenue models are explained in detail in Table 1.
Table 1  Open source revenue models (Rajala et al. 2006)

<table>
<thead>
<tr>
<th>Revenue Model</th>
<th>Description</th>
<th>License types</th>
<th>Revenue sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support selling</td>
<td>A for-profit company provides support for a software that is distributed free of charge.</td>
<td>Any</td>
<td>Revenue comes from media distribution, branding, training, consulting, custom development, and post-sales support for physical goods and services.</td>
</tr>
<tr>
<td>Loss-leader</td>
<td>A no-charge open-source product is used as a loss leader for traditional commercial software, i.e., the software is made free by hoping that this will stimulate demand for a related offering the company has.</td>
<td>Varies</td>
<td>Complementary offerings, e.g. other software products</td>
</tr>
<tr>
<td>Widget-frosting</td>
<td>Companies that are in business primarily to sell hardware can use this model for enabling software such as driver and interface code. By making the needed drivers open the vendor can ensure that they are debugged and kept up to date.</td>
<td>Any</td>
<td>The company’s main business is hardware. This is quite similar to the loss-leader model.</td>
</tr>
<tr>
<td>Accessorizing</td>
<td>Companies which distribute books, computer hardware and other physical items associated with and supportive of open-source software.</td>
<td>Any</td>
<td>Supplementary offerings</td>
</tr>
<tr>
<td>Service enabler</td>
<td>Open-source software is created and distributed primarily to support access to generating revenue from consulting services and on-line services</td>
<td>Any</td>
<td>Service fees</td>
</tr>
<tr>
<td>Brand licensing</td>
<td>A company charges other companies for the right to use its brand names and trademarks in creating derivative products</td>
<td>Strong reciprocity</td>
<td>Copyright compensations</td>
</tr>
<tr>
<td>Sell it, Free it</td>
<td>A company’s software products start out their product life cycle as traditional commercial products and then are converted to open-source products when appropriate.</td>
<td>Alteration of license type</td>
<td>Initial revenue from software product offerings converted into other models, e.g. the loss-leader model</td>
</tr>
<tr>
<td>Software franchising</td>
<td>A combination of several of the preceding models (in particular “Brand Licensing” and “Support Sellers”) in which a company authorizes others to use its brand names and trademarks in creating associated organisations doing custom software development in particular geographic areas or vertical markets.</td>
<td>Strong reciprocity</td>
<td>The franchiser supplies franchisees with training and related services in exchange for franchising fees of some sort</td>
</tr>
</tbody>
</table>

As we can see from the table above, there are multiple business and revenue models for business with OSS. The key issue is that traditional business models and revenue logic is not directly applicable in CoopWorks settings when assisting the development and sustainability of the software. Understanding of open source business is important, in order to encourage, rather than limit the business activities.
CoopWorks user and developer community

It might be possible to develop and support CoopWorks version(s) in the Internet as a traditional open source software project. Several characteristics are common to many OSS projects. The generic OSS development process (Feller and Fitzgerald 2002)

• is parallel, rather than linear
• involves large communities of globally distributed developers
• utilises truly independent peer review
• provides prompt feedback to user and developer contributors
• includes the participation of highly talented, highly motivated developers
• includes increased levels of user involvement
• makes use of extremely rapid release schedules

There are many advantages in open source development, such as involving many innovative programmers, allowing the free development of modules or even new versions, potentially increased quality in the software, and better involvement of the users (see more in e.g. Raymond 2001). However, clear definitions, leaders, and tasks have to be provided. The original developed of the software must provide already ready code for the community to test and improve. Open source leadership must provide an initial vision, communicate clear procedures and be perceived as fair. If a participating party is not happy with the actions or behavior of the project leader(s), that party can challenge those actions or even fork the project, using the code base to clone the project. However, poor leadership, or lack of it, usually ends up in diminished activity in the project. The development and activity of the community is not easily attained, and FAO and other participating organizations should actively support the community.

Academic community

During the inception mission we established contacts with the University of Nairobi. The key results were:
a) Relatively high level of OSS expertise available in the computer science department;
b) Enthusiasm and support available at the department of entrepreneurship, of which is a concrete example a forthcoming conference which has dedicated ICT in agriculture track. It seems that further collaboration with the universities of Nairobi is advisable.

Discussion and conclusions

Dependence on proprietary software products may cause several disadvantages in developing countries, such as diminished adoptability of the software product due to high costs and inflexibility, and piracy problems.

Implementation of software on open source platform, or total platform independence, that makes it possible to run the software on many operating systems as well as server technologies, also makes software easier to be adopted and customized to various needs. In addition, an open source platform compatible version would benefit more of the advantages of the open source approach. Indeed modularity, reusability and architectural independence further enable the development of new modules, and their inclusion to the main project, also in the case of CoopWorks.

When the OSS community has a significant role in the development and diffusion of the software, it is important to focus on a single software version. Just one version of the software, which could be run on all platforms, means also that forking of the project to many versions according to specific needs would be avoided, and the developer community would include more potential developers. Versioning of the software product at divides resources and creates market uncertainty to potential uptakers of the software.
In this case study it was apparent that in Kenya there was a need for a wide variety of actors, because no single organisation could provide all the necessary services needed for a co-operative to start computerisation to the full use of CoopWorks, nor was this the purpose of the project. Open source approach provides many kinds of business opportunities, which are also beneficial to the project, and should therefore be encouraged. Support for commercial activities by the government and by the UN can be considered very beneficial to Kenya in general, since the country suffers from loss of highly educated people due to lack of opportunities. However, traditional business models and revenue logic is not directly applicable in CoopWorks settings, and open source development and business should be properly understood. Also, birth and development of an active user and developer community offers many advantages, and are also clearly the aims of the project. However, development and activity of the community is not easily attained, and the project coordinator and other participating organisations should actively support the community.

The management should be actively involved in developing the user and developer community, and the business ecosystems. In fact, it would be crucial that the software project would have a local, as well as international leader, who would in a neutral, responsible and sustainable manner lead, manage and support the project. In addition to this leader, the project should involve other governmental, non-governmental and business organisations.

In order to ensure long term viability and sustainability of the software project, as well as enhancing the creation of local business environment, it would be necessary to encourage, rather than limit or hinder, all related business and different business models. Also the special characteristics of open source related business should be kept in mind.

It is also important that the project would emphasise the further productization of the software, in order to improve its adoption by both ICT companies as well as co-operatives. Productization could include activities such as development of business, investment and implementation processes, guidelines and other material to support these activities; development of the CoopWorks web site, and the user/developer community etc.

References

Wright, B. “OSS Could Be Africa’s Answer”. Linux Insider (www.linuxinsider.com), article dated 06/17/11.