Topic 5: FOSS-specific business knowledge and skills

Brief description of Topic

While the previous Topic 4 transmits the knowledge that is necessary to build any business at all, this topic deals specifically with FOSS businesses. Though FOSS businesses largely work along the same rules as other businesses, there are some things that work differently in FOSS, and some challenges that are specific to working with this type of software, many of them based more on the perceptions of potential clients than on facts.

The knowledge in this topic will help participants to clearly see the specific aspects of doing business in FOSS, and will help them to address the particular challenges of this field of work.

This topic is complementary to Topic 4.

Objectives of the topic

At the end of this topic, training participants will

- understand the main differences between running a FOSS business and running a proprietary software business, as well as non-software types of service businesses
- be familiar with the strategies of cooperation and coopetition that are typical for FOSS
- know the specific challenges in marketing FOSS, and how to overcome them
- have a basic understanding of the role of innovation in FOSS business.

Table of Contents

Detailed outline of sections

Day	Session	Duratio n
	[FIXME]	
Total		**9.5 hrs**

1 Section 1: How is FOSS business different from other types of business?

Contribution to objectives of topic:

1.1 Revenue streams

The most obvious difference between doing business with proprietary software and doing business with FOSS is that a FOSS company will usually not make money by selling licenses. (Business models based on dual licensing are an exception here.)

However, this difference is not as significant by far as it would appear. For example, proprietary packaged software in the United States accounts for less than 10% of software developers employed.[1]

Note that when re-selling licenses for proprietary software, most businesses only make single-figure profits (i.e. below 10% of the license price). The rest of the client's money ends up with the original seller of the software, quite frequently leaving the country and the continent for the US or Europe. The buying power of this money will effectively be lost for your business.

The absence of licensing costs has various effects:

- your business is able to draw on a large supply of high quality software components free of charge. You will be adding value by configuring those components and making them work together, rather than by developing them from scratch.
- when your client's choice is between a solution based on proprietary software and a solution based on FOSS, the lack of license fees means that almost all of your client's budget can go towards paying the

1 see http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf

services you provide. This is money which your business will keep, as opposed to sending a substantial share of it abroad to cover license fees.

1.2 Targeting activities to add value:

When working with FOSS, you can focus all of your business' resources on those activities that add value to your product or service. Since you are able to build on existing components, you aren't forced to spend time and money on developing those parts of your offering which brings in the most revenue.

If you are putting together a FOSS solution, you will usually be able to draw upon a large number of ready-made libraries and software packages. You will put your expertise to work in at least some of the following fields:

- designing the solution
- selecting the appropriate software packages
- making them work together
- customise them for your client's needs
- setting up the new solution
- managing the client's migration to the new solution
- training users and system administrators to work with the new solution
- ongoing maintenance and support contracts

These activities translate into at least the following business models:

- Training
- customisation
- Technical support, maintenance
- Monitoring, updating
- Service level agreements (SLA)
- custom software development, often centered around a FOSS project in which your company may have found an important role in the course of acquiring experience

All of these services can be provided on a one-off basis, or can be sold to the client as a subscription.

The need for FOSS-related services is an important business opportunity for you. It is also an important opportunity for local economic development.

1.3 Vendor-client relationship

In FOSS, all important components of a program - executable code, source code and documentation - are usually publicly available. This means that clients have the possibility to inform themselves about the software they might need, and gives them the opportunity to better judge the solutions that your business is offering. This allows for a much more equitable relationship between vendor and client.

This relationship in turn means that the client is more likely to get exactly the solution she is asking for. Ideally, this leads to greater satisfaction on the client's part, laying the basis for a long-term business relationship.

In sum, it might be helpful to consider yourself not as a software provider, but as a service business. Much like a car mechanic or a hairdresser, you are selling your time rather than any particular product, and you get paid at the level of your skills.

1.4 Second subsection

Exercise 5.1:

Spend five minutes browsing the following websites:

- http://support.novell.com/linux
- http://www.arimaan.com/technology/opensource.html
- http://www.siriusit.co.uk/
- http://www.lomboz.org/web/guest/support

(Disclaimer: these URLs are provided for illustration only. The vendors they point to are in no way endorsed by trainers or program managers)

1.4.1 Case study 1.2:

1.5 Third subsection

Exercise 5.1:

Exercise 1: Look at the case studies in Topic 2 and identify the different revenue streams of each company described there. [FIXME: training participants may add those revenue streams to this module as information] [30 min]

Exercise 2: A client asks you to build a mail system for his enterprise, which employs ten people, including a mail server and desktop clients. He wants to be able to administer the easier aspects of this system himself. Go online to identify which existing FOSS components you can use to build the system. [30 min]

NOTE: at the end of each section, there must be at least one of the following:

- **exercise / assessment**
- **Case studies or essential industry information**
- **group discussion**
- **questions for contemplation and reflection**

2 Section 2: FOSS communities

Perhaps the most important aspect of a FOSS community is that it is a reservoir of knowledge and skills, which community members are usually happy to pass on to others. Communities are a place of learning, and you will learn in proportion to the degree to which you get involved in a community.

This learning takes place on different levels.

On the technical level, communities function as informal apprenticeships. New entrants choose their field of interest, and make their best efforts to contribute. More experienced community members typically provide feedback and advice to the new entrants on how to improve their contributions.

But people who are active in FOSS communities also find that they are excellent places to learn about teamwork and cooperation. Since community participants are usually bound together by little else beyond a shared interest, it is essential that every voice is heard and conlicts are resolved amicably, while keeping everyone's eyes firmly on the overall goal. These are practices that a good manager will also carry over into her business, making it a more agreeable place to work. This in turn makes it easier to retain qualified staff.

Most businesses will probably get the greatest advantage out of their participation in FOSS communities when this participation revolves around technology that is important, but non-differentiating, i.e. not something that sets their business apart from everyone else's.

There are two advantages for you here. If your business is in offering services for the Plone CMS, you have an interest in seeing that system improved. Of course, this will also benefit your direct competitors (i.e. other people offering services for Plone). But it will increase the overall size of your market, since a better CMS will attract more clients.

The other advantage becomes effective if you are offering an add-on to the community's technology, and generate revenue from providing this add-on to your customers. Again, with a better CMS (to stick with the Plone example), the size of your market will increase. In addition to bringing in more revenue for services related to the basic CMS, you will also more frequently be able to charge for distributing the add-on.

For this to work, the add-on does not have to be proprietary. Even if it is licensed as FOSS, you may simply choose not to distribute it to anyone else than paying customers. While those customers are in theory free to pass your add-on to other people, they will probably not do so because it would require them to make an extra effort. Also, the recipients would probably need your services to fully benefit from that add-on -- so even if your customers do pass it on, this means more business for you.

If you provide a FOSS solution to a customer, that customer can also be considered a part of the community around the software involved. This means that there is a good chance that your customers can become relatively well informed about the capabilities and benefits of a solution - if they choose to invest the necessary time and effort. If they do so, they might of course start relying less on your services and more on their own work, in combination with support from the community. But experience shows that this is not usually the case. Rather, most of your customers will choose to focus on their own core business, and continue to ask for your services in supporting the solution.

Exercise/Question/Case study/Group discussion 5.2:

[FIXME]

3 Section 3: Competition, cooperation – coopetition

When you base your business on proprietary software, your options will be limited. There will be other companies providing similar products and services, and you will be competing with them.

When you base your business around FOSS, the situation is different. There still will be businesses offering products and services similar to your own. But you will be able to choose from a much broader spectrum your ways of dealing with these other companies.

Your attitude towards other FOSS companies which are active in your region will fall somewhere between the poles of competition and cooperation. This middle ground is sometimes called "coopetition" - competition and cooperation at the same time.

Coopetition can take many forms. One of these forms is outsourcing inside a network of partners. At least at the beginning, your business will be small. When faced with a large and complex task, you will often not be able to do everything yourself. The solution is to share the work with other firms with different specialisations. You will be letting others do things that you can't (profitably) do yourself. This gives you the ability to take on larger tasks and deals by sharing them with trusted partners from the network

Another (and often related) form of coopetition is sharing costs among members of the network. This can mean sharing marketing expenses and costs for PR work, and building a common image for the network [see www.zeapartners.org]

Taken together, higher revenue (from larger deals) and shared costs lead to higher profits for all partners.

A business network will also be vital given that many potential clients remain underinformed about FOSS and may be sceptical towards this type of software. This means that FOSS businesses must generally make an effort to increase the general acceptance of FOSS. If they succeed in this endeavour, they clearly stand to profit.

FOSS business networks take different shapes. Some are centered around a certain piece of software. Others are associations of businesses with a general interest in FOSS. Some associations such as FOSSFA [http://fossfa.net] are engaging in broad FOSS advocacy. This latter type of association is important because it helps to create a climate that is more welcoming towards FOSS. Such advocacy will prepare the ground for your business, and make your clients more receptive to the FOSS solutions you propose. This includes potential customers from both the public and the private sector.

4 Section 4: Marketing FOSS

4.1 Clients and markets to target

Choosing which clients and markets to target is a strategic decision for your company. Your choices include:

- other firms, both large and small
- public bodies
- local representations of international organisations
- educational institutions: schools and universities
- individual users

Each of these groups of client will have different specific requirements. Of course, these requirements also vary with the country, region and city or town your clients are based in.

The better you get to know the requirements of your clients, the greater your chance of satisfying them. For this reason, many FOSS companies specialise in serving only one or two of these groups.

4.2 Overcoming barriers to adopting FOSS

Below, we discuss oft-cited barriers to adopting FOSS, along with their respective solutions.

4.2.1 Availability of support

YOU are the solution. You can build a business on providing support to FOSS users, especially larger organisations. The fact that FOSS is often well documented, with vibrant communities around many programs, means that you will find it relatively easy to build up the necessary skills.

4.2.2 Availability of applications

There is an enormous bandwidth of FOSS applications, serving almost every purpose. There are very few areas where there is no adequate FOSS application available to perform a certain job.

However, many people have grown attached to proprietary applications they have long been using. When preparing a migration, it is therefore essential to reassure them that they will have available the applications they need, and that they will be able to use those applications. The former may be accomplished by showing clients in detail which FOSS application will replace which proprietary program. The latter is a matter both of selecting user-friendly applications, and of offering adequate training to users.

In larger organisations, there usually are three approaches to replacing proprietary applications. These can be combined as necessary:

1. find and deploy a sufficiently functional FOSS replacement.
2. for applications where there is no FOSS equivalent, one option is to make them available on the internal network via a terminal server.
3. continue to run the proprietary application in a virtual machine.

4.2.3 Software quality:

FOSS very frequently is very good, since in theory everyone is able to inspect the software, find errors, and report or fix them. Yet quality may vary, especially for projects with small communities. The overall variation in quality is not greater than for proprietary software.

4.2.4 Legal concerns:

Some clients may worry about becoming exposed to legal claims by using FOSS. In fact, the legal risk associated with FOSS is no greater than in the case of proprietary software. Usually, it is even smaller, because your client does not have to worry about acquiring software licenses.

It is your task to ensure that you adhere to the licenses of the FOSS packages you are using when integrating a solution. This should be a standard item in your contract with your client.

These legal worries also represent a business opportunity. You can offer, for a fee, to indemnify your client against legal claims made in relation to the solution you're selling.

4.2.5 Untold fears:

A client's CIO may fear that his importance in the company will shrink in line with IT costs, as budgets are adapted. This will usually not be the case. On the contrary, when costs do shrink significantly, the CIO will be able to claim the managerial merits for the savings.

Users may sometimes believe that the absence of a license fee for the software means that this software is low-end, and by extension that their work is not taken seriously. Here, it is very important for all those involved in the deployment of a FLOSS application or solution to work *with* the users rather than against them. A good way is to talk to as many users as possible to hear about their IT needs. Your time will be well invested, since users feel that they are being listened to. Accordingly, they will be much more open to the changes you bring to their work environment.

4.3 The TCO debate

TCO stands for "total cost of ownership", meaning all costs associated with acquiring, installing and using a piece of software. This loosely defined concept covers a wide range of items, such as license fees, installation costs, and maintenance costs.

There is an argument about whether TCO is lower for proprietary software or for FOSS. While TCO varies from case to case, FOSS will generally have an advantage when long-term costs are figured in.

Somewhat counter intuitively, the fact that FOSS is available free of license costs does not greatly influence its position in the TCO debate. Even for proprietary software, license fees make up only a small proportion of total costs over time. This is especially true when your FOSS business is working in an environment where much of the proprietary software that is being used is unlicensed, and has been acquired at a price near zero. For large deployments, it is also quite common for large proprietary vendors to use a loss-leader strategy, giving away software licenses e.g. for the operating system for (nearly) no money in order to gain customers for applications building on that operating system. For these reasons, license fees often fail to work as a convincing argument for FOSS.

It is common for customers to move to move to new applications, be it because they offer better functionality, or because a proprietary supplier has disappeared from the market. Since most proprietary applications store data in their own proprietary format, it is usually very difficult for the customer to convert his data to a new format.

FOSS avoids this problem by using open standards - ways of sending, receiving and storing data that are publicly documented, and can be implemented by anyone. This has two important consequences:

- it is easy to replace one component of a FOSS solution with another.

The customer is not bound to specific applications, but can simply choose the ones that work best.

- the customer avoids being locked into a particular software or file format, giving him greater flexibility in choosing suppliers.

This usually means substantially lower costs over the long term. Critically, it also means that the customer does not depend on a particular application from a particular supplier to access and manage his data. The customer effectively gains control of his own IT infrastructure.

The TCO argument is often used by proprietary vendors, using questionable data to argue that while FOSS may be cheap in the beginning, it costs more over the long term. In reality, the opposite is usually the case: E.g. a migration to FOSS will carry about the same costs as a migration to the next generation of proprietary software, but once the migration is over, IT costs tend to become substantially lower.

4.4 Making the case for your FOSS solution

There are a few steps you should follow when working with a client, in order to come up with a solution that meets the client's needs:

1. Identify the client's needs. Don't just rely on his explanations; try to observe first-hand how his company or organisation works, so you will get a better idea of the problem you're trying to solve.
2. Find FOSS products that meet your client's needs.
3. Compare them to proprietary alternatives, on functionality quality and cost.
4. Select the best products, and reject the others.
5. Try to find unbiased data to balance the marketing hype.
6. Be comfortable with your decisions, and be ready to present them to your client in a convincing manner.

Exercise 5.4:

Exercise 5.4.1: For an illustration of the independence afforded by FOSS, read through the case study on the German city of Munich's migration to GNU/Linux. [http://www.osor.eu/case_studies/declaration-of-independence-the-limux-project-in-munich] Identify Munich's reasons for migrating, and the benefits the city hopes to obtain.

Exercise 5.4.2: Read through Mark Taylor's article on "The true cost of migrating to open source" [http://www.zdnetasia.com/insight/software/0,39044822,62054142,00.htm]. Imagine yourself in discussion with a sceptical client: how do you convince him that a migration to FOSS will be the better option in the long run? Among other things, your client will confront you with the view that "FOSS has a higher TCO".

5 Section 5: Innovation in FOSS business

5.1 Innovation vs Commoditisation

In IT, commoditisation happens constantly - a program which used to be special in some way will attract competitors, both FOSS and proprietary, until there are a number of programs functioning in near-identical ways. Customers are easily able to replace one commodity program with another.

But software is developed in a modular way. At any point in time, parts of the value stack (the operating system, middleware, applications, services) are commoditised, while others are not.

For businesses, the problem of commoditisation is that there usually is strong competition to sell near-identical products. This means that whatever profits can be made will be relatively small. The opportunities for attractive profit margins move upmarket, meaning that companies need to keep innovating in order to preserve their profits.

It could be argued that for most purposes, operating systems have become a commodity, since most users will be able to satisfy their needs using any end user-oriented operating system.

While prices of operating systems vary slightly, these are not the items that usually generate profits for the company selling the system. It is the programs that run on top of the operating system which bring in the money, such as office software.

Much more relevant for FOSS companies, users also require professional services, which are also far from being commoditised in most regions, meaning that a good service provider can expect relatively high profit margins.

5.2 Use of FLOSS in an innovation strategy

"Innovation is a driver of economic growth, productivity, job creation and rising living standards."

"Innovation also promotes ICT competitiveness; in turn, competition leads to better products, improved consumer choice and, ideally, greater ICT uptake."[2]

Most products and programs - FOSS or otherwise - are not innovative, in the sense of bringing about a new way of doing something. Radical innovation is also extremely rare.

Most innovation happens in an incremental fashion, by taking something that already exists and making it slightly better.

In proprietary software, this is something that can only be done by the person or company holding the copyright in that particular piece of software.

In FOSS, the possibilities are far greater. Here, anyone can take the software and improve it. While this does not mean that there is more radical innovation in FOSS, it means that there are huge opportunities to build small improvements on top of strong, existing foundations. This sort of innovation is well within reach of individuals and SMEs without big R&D budgets. It also means that if the idea fails, the innovator's lost investment is usually small enough to not do substantial damage to her business.

5.3 Open innovation

"Open innovation" means that individuals or firms work together to come up with an improved program or product. This is not exclusive to FOSS, but it does represent a clear departure from the usual practice, where each firm tries to keep its innovations secret from its competitors.

Innovation happens

- within FLOSS projects
- on products based on FLOSS
- through percolation from proprietary development processes
- and in the process itself: open innovation

Open innovation works particularly well in software, since source code can easily be shared, and the necessary tools (computers, compilers etc) are widely available. Beyond this merely practical point, there are clear incentives to work together in many situations:

- sooner or later, a proprietary offering will be re-implemented in FOSS
- the costs of software production can be reduced through collaborative engineering

2 Source: European Task-Force on ICT Sector competitiveness & ICT uptake, WG on innovation in R&D, manufacturing and services http://ec.europa.eu/enterprise/ict/policy/taskforce/wg/wg3_report.pdf

- opening the source code is the best way to maximise the potential for collaboration
- when competition becomes pointless, collaboration is a good way to keep innovating, thereby improving the software upon which all competitors base their business

FOSS provides a ready-made legal framework for cooperation. If you develop something, you can share it with others, using a copyleft license to keep competitors from appropriating your work.

The basic rules for developers in FOSS are:

- reuse instead of reinventing the wheel
- contribute patches and enhancements - don't try replacing the program you're contributing to with something of your own invention
- communicate with your peers

Materials

Please provide links to the following materials:

- PDF version of this topic
- reference documents
- Teaching notes
- Lecture slides
- Handouts

Sources

Significant parts of this topic are based on Francois Letellier's teaching materials for FOSSBridge Vietnam. © Francois Letellier 2008, published under Creative Commons Attribution Share Alike License 2.0.