

Sustainability of FLOSS-Based economic models

II Open Source World Conference, Malaga

03/15/06

Carlo Daffara, Conecta
European Working Group on Libre Software

- While a great deal of research has been done on motivations and roles of individual developers, less has been done on why private firms do work with FLOSS at all
- A recent research (Bonaccorsi, Rossi 2004) shows that firms are less bound to social and philosophical reasons, and are interested in reaping the technical and economical benefits
- The main questions are:
 - Is FLOSS development compatible with business models?
 - Are these models different in respect to traditional SW models?
 - Are they stable/sustainable?

- This research activity was partly funded by the IST programme of the EU, in the COSPA project (“consortium for open source in the public administrations”) that aim at analysing the effect of the introduction of open source and open data standards for personal productivity and document management in European Pas
- More information at <http://www.cospa-project.org> , where you can find additional material like a complete CC-licensed training package for OpenOffice.org

- “The GPL effectively prevents profit-making firms from using any of the code since all derivative products must also be distributed under the GPL license” (Evans, D., in “Government policy toward open source software”, R.W.Hahn, editor, AEI-Brookings JCRS)
- “[..] the aim of free software is not to enable a healthy business on software but rather to make it even impossible to make any income on software as a commercial product” Thomas Lutz, Microsoft representative at Tunis WSIS

- ...HP reported Linux-related revenues of more than 2.5B\$ in 2003, and reported in July 2005 to have sold its millionth Linux-only server
- Total Linux market share for 2008 is estimated by IDC to reach 35.7B\$; on 2004 28.3% of server sales were Linux related, and it is expected to reach 37.6% of the market in 2008
- On smaller examples, RedHat reported quarterly earnings of 63M\$ as of Sept 2005
- And these are only for Linux! Think about Jboss, SleepyCat, Zend...

- In the European Working Group on Libre Software (Barahona, Daffara 2000) we introduced a categorization based on the distinction between “internal” funding (related to reduced cost, or higher technical value) and “external” funding, related to service sales
- We also included public funding as a potential model
- Is there a need for public funding?
- Yes, if there is a clear market failure...
- ...and several vertical areas are experiencing exactly that:

The need for public funding

“the 1999 PITAC analysis of high end computing noted that suppliers of high end systems were becoming an ever-smaller fraction of suppliers of high end systems were becoming an ever-smaller fraction of America’s information technology-driven economy. Indeed, the absolute size of the high performance computing (HPC) market seems stable to declining. The Committee’s technical analysis revealed that while there were a number of high end applications ripe for exploration, the field was in need of substantial innovations in application-development software, algorithms, programming methods, component technologies, and architecture.” (PITAC report on open source for high performance computing, 2000)

From public funding to internal funding

- The same failure exist in other areas as well
- Several examples exist, all related to a failure or inability of the commercial market to create a suitable alternative
- In general, the packaged software makers are using the “majority” rule of design:
 - develop code for the majority of users
 - if possible, make the program extensible, so it can accommodate further uses without added efforts from the original developers
- but this explicitly avoids all “niche”, vertical sectors where there is no return on investment...
- ...including all “innovative” and untested areas



Why SystemC?

- EDA developers
 - Small market (compared to SW)
 - Base a new language on an existing widely used language
 - Rely on existing tools
 - Focus on EDA specificity
- Common interest from several companies
 - Contributions from existing internal technologies (Synopsys, CoWare, Adelante Technologies (was: Frontier Design))
 - Now extended to others

The “needed improvement” model

- Most of the software is internally developed, and is a significant cost
- A way to reduce cost is to leverage existing FLOSS packages, and just adapt or modify it to match the internal needs
- If the FLOSS project adopts the patches, the cost of maintenance is reduced significantly
- This justifies the large number of companies that contributes back to the FLOSS pool; for example, in Henkel, J, “Free disclosure of innovations: the case of embedded Linux”, Oxford workshop on libre/open source software, 2004:

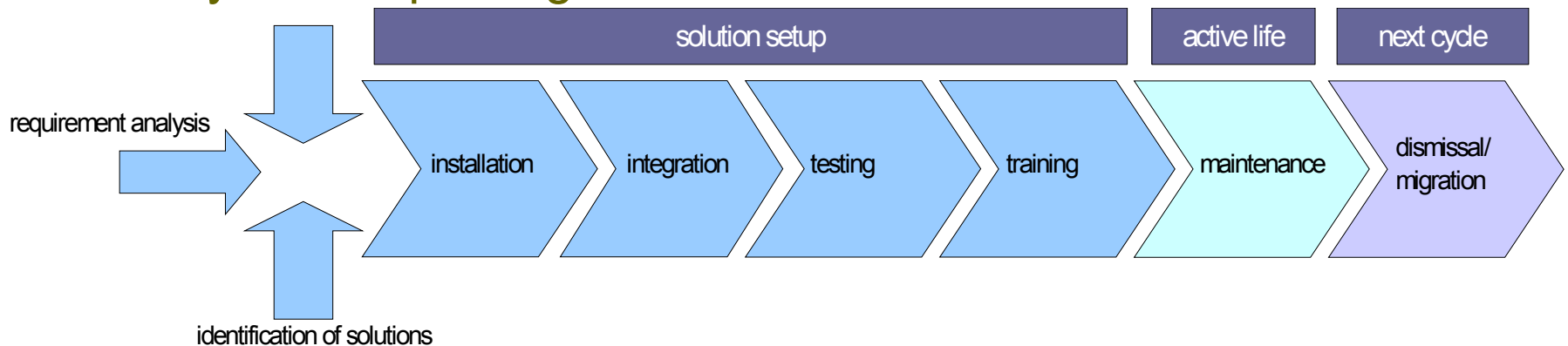
<i>Type of firm</i>	<i>Share of code revealed</i>
Device manufacturers	42.30%
Component manufacturers	58.80%
Software firms	57.50%
universities	90.00%
hobbyists	93.30%

The “needed improvement” model

- This foster the creation of “collaborative consortia”, groups of companies or public entities cooperating in the development of artifacts that are unsatisfactorily or uneconomically provided by the market, and that use (or reuse) FLOSS as a basis
- One of the most successful one is the SAKAI group, that is developing a course and faculty management system for universities
- another example is KUALI (financial information system):
- “... The inflexibility of some of these general purpose systems, particularly for the financial module, stems from adapting vended software for corporations to the very different needs of higher education. Many campuses with mature home grown systems are finding that the new packaged application have actually *reduced* rather than enhanced the functionality needed by administrators to manage colleges and universities.” (Kvavik, R.B. Katz R.N. “The promise of performance of enterprise systems for higher education”

Business models: a taxonomy

- The adoption process of an open source solution is similar to that of any ICT infrastructure; the difference is in the possibility of testing of all the components prior to the selection process, and a greater flexibility in adapting different tools and infrastructure
- On the other hand, there is less information on all but the most widely known packages



Business models: a taxonomy

- Business models based on the adoption process:
 - Software selection (if Off-the-shelf components are used)
 - Installation
 - Integration
 - Technical suitability certification
 - Legal certification
 - Training
 - Ongoing maintenance and support contracts
 - At the end, migration from the old system to the new one
- Business models based on constraints or externalities;
 - Twin licensing
 - Time-Decaying licensing
 - Mediation services
 - Commercial-on-open
 - Loss-leader model
 - Custom developments
 - Ancillary markets

- a multi-step phase, that starting from the identified needs and a knowledge of the software market selects a combination of packages that minimize the amount of code that needs to be developed
- This minimization process is complex, taking into account not only the technical characteristics of the software being considered, but also must provide an evaluation of the “liveliness” of the OS project, the probability that its development will continue, and the availability of consultants and documentation

- Companies that want to offer software selection consulting services must dedicate a certain effort just in monitoring web sites and mailing lists, and extract from there information on new versions or new packages; this effort can be estimated in 1 man/hour per day for limited segments (for example, only java enterprise middleware) to 5 man/hours per day for many different software segments
- Analysis using standardised models (OSMM, BRR) requires 1-3 man/days per evaluation, that needs to be repeated when a new version is available; this means that it is necessary a substantial initial investment in terms of effort to create a suitable database

- A very common support activity in the OSS community is that of installation; this comes from two different aspects: the great modularity of software (that forces the installation of many, different components to create a working system) and the relative unavailability of sophisticated software installers, common in the commercial world
- The availability of package installers based on the RPM and DEB augmented with dependency maintenance systems have greatly reduced the complexity of installation, now mostly related to the modification of the suitable configuration files to adapt the installation to a specific ICT environment
- In the future, probably virtual machine images will be used instead to easily replicate complex installations

- Another of the most common steps in OSS-based consulting is the integration step, that relates both to the specific configuration step necessary to “fit” an open source component in an existing structure, and to the custom development necessary to add the missing functionalities or correcting the incompatibilities
- Integration may require a substantial effort for large scale projects, with a relatively large amount of custom coding or the integration of commercial components if no other choice is possible. This variability is the reason behind the strong push towards standards (both de-facto and de-jure) that is the simplest way to reduce interfacing cost between disparate software components

- This is mostly done by integrators and external consultants, and may come in two shapes: certification of adherence to an international standard (for example security or quality standards) and certification of suitability for a specific environment
- The integrator or certifier provides an insurance that the software package complies with a specified set of rules, and is legally liable for such compliance. Limited scope certifications, like security assurances, are quite within scope of SMEs, while large scale quality assurance of components is quite difficult to attain if the open source project itself does not have an in-place explicit mechanism for project management

- The first area is related to the mixing and correct use of components, that may have different licenses and different restrictions. While more than 70% of the open source code is actually released under the GPL, more than 50 other licenses exist, and some fundamental components are released under a non-GPL license
- Patent and IP certification provides a form of “insurance” against third party claims on software patents or other copyrighted material that may be in the OSS used in a project; as any insurance form, it is quite demanding in terms of monetary funds, as patent claims may give rise to multi-million Euro lawsuits

- Training is another of the easiest business models, as many open source projects do not have an official, sanctioned training process that is comparable to that of commercial companies, that may create specific training and certification programmes
- Training is usually personnel-intensive, and requires some effort for the creation of the initial training material to be used during the courses. A good estimate of work needed is that it is necessary to invest around 3 to 8 hours of course material preparation for each hour of training delivered
- The simplicity of the model and the fact that it does not require software development means that it is quite easy for established training companies to compete for offering such services; also, the largest OSS project also usually have an official training programme (for example JBoss, Linux distributors RedHat and Novell/Suse)

Business models – Maintenance/support contracts

- Support contracts usually are time-based (the most common is a contractual period of one year, renewable) and level-based. Levels are commonly three (corresponding to “bronze”, “silver” and “gold” support services), with varying degree of guaranteed service, or “tokens” can be used to buy services in a flexible manner
- While it is reasonably easy for an SME to offer standard support services, 24/7 offerings may require a slightly larger personnel base to guarantee coverage under every circumstance
- The support model is used by many companies that turned a commercial package (not completely successful in the commercial market, or unable to completely fulfill its market potential) into an open source one; the underlying idea is that the authors of the code are supposed to be the most qualified experts for support it

- Most migration services are based on software packages that help in automating the migration (for example of user configurations), or on pre-configured “packages” of OSS that provides complete substitutes of proprietary environments
- Examples: groupware services, security, edge services, structured desktops
- for some large scale effort may require coordination among different companies, offering coordinated service (for example, one specialized in porting custom code, one in migrating mail services, etc.)

- Twin licensing is used by companies that want to profit from the companies that want to use or leverage an open source package without standing the redistribution conditions of the OS license
- The model can be effectively used only for source packages that needs to be linked in with the code for maximum efficiency or because there is no common protocol for data exchange
- Twin licensing requires some specific legal and community aspects to be handled; for example, patches or modifications from external contributors require an explicit author acknowledgement of both licenses, and requires an accurate management of the border between the commercial and open source aspects of the project

Business models – Time decaying licenses

- Time-decaying licenses where a software artefact changes license with time or with some specific event (for example, the release of a new version of the code)
- The first known example of this model was the Alladin Ghostscript postscript interpreter, and recently some security companies provide up-to-date security signatures to paying customers, and release them under a public license after some days
- This model is especially suited to rapidly changing software or other material (for example, security and virus signatures) and less practicable for software, because the old version becomes a basis to create an improved product that may be competitive with the one under the commercial license

- Are based on the fact that for companies it is difficult to interact with sparse communities like some OSS projects
- Mediation services provide a sort of a single point of contact, that gathers information from the developers, mailing lists, forum and such and forwards requests and bug-fixes back
- Usually these mediation companies try to contact directly the developers, or to find support companies that demonstrate experience in the specific package; after development, they add some certification and integration effort to deliver a single package to the customer

- One of the simplest model for software companies is selling a proprietary software package on an open source one. It may be simply a matter of running platform (like having a commercial package running on Linux) or it may leverage an open source project with some commercial module
- Any company that plans to follow this model should devote some effort to track the evolution of the OSS platform, and somehow participate (for example, with an active participant in the mailing lists of the project). This has the double advantage to provide an insight into the evolution of the platforms and new, potentially useful features, but also to be “good citizen” of the OSS project

- Effort is invested in an open source project to create or extend another market under different conditions
- For example, hardware vendors invest in the development of software drivers for open source operating systems (like Linux) to extend the market of the hardware itself
- Other examples are related to the establishment of a platform or a specific protocol; for example the Eclipse project was extremely successful in creating a large ecosystem of tools and projects that complement and enhance it

- The offering of custom coding on an open source project
- There is usually a form of specialization on a single project or class of projects (for example, device driver development or open source-based J2EE systems)
- The company that wants to use this model should add to the traditional model an activity related to tracking the evolution and roadmap of the project on which it is specializing, in a way similar to that described in the previous commercial-on-open model

- Ancillary markets are those that arise in an indirect way; for example, the lack of proper documentation for a large number of OSS projects led to a substantial market opportunity for editors (like O'Reilly, that specializes in books and guides on popular open source tools)
- Many are surprised of how much money can be made by merchandising...

The mix model - examples

- The Linux distributor model (like RedHat, Novell/Suse or Mandriva): a combination of software selection (the choice of packages on the distribution CDs), integration, technical suitability certification; they also provide training (through certification courses and training material) and support. With some specific contracts, they also provide legal certification (in the form of indemnification contracts)
- The wholesale vendor (like IBM, HP, Sun): this model combines many different aspects, like loss-leader (paying for development of drivers to sell hardware), consulting, integration, legal and technical certifications, training and maintenance. Also, almost all the vendors add to this a line of proprietary software that complements the open source offering
- The OS consultant: this model focuses on the consulting activities, like software selection, and outsources the parts that require a high personnel effort, like training and custom development; the consultant provides the “end points”, offering both the initial selection and the final technical certification.

Vertical specialization

	pkg1	pkg2	pkg3	...pkg n
Software selection				
Installation				
Integration				
Technical suitability certification				
Legal certification				
Training				
Maintenance and support				
Legacy migration				

The most common form of specialization is “vertical”, that is it covers most or all of the models across a limited number of packages. For example, vendor specializing in a single package, or offering “turnkey solutions” like prebuilt servers

Horizontal specialization

	pkg1	pkg2	pkg3	...pkg n
Software selection				
Installation				
Integration				
Technical suitability certification				
Legal certification				
Training				
Maintenance and support				
Legacy migration				

The specialization process leads then to an “horizontal” model, where a company performs a single activity on many packages (for example, a company specializing only on training or certifications).

The mix model

- Most “first movers” were motivated by strong philosophical/social motives...
- ...that later changed to pure economic incentives
- Most started as consulting firms doing many little tasks, because it is the easiest model to follow; but it does have several disadvantages:
 - the small customer has limited willingness to pay
 - the large customer wants to internalize knowledge, so it does have to pay only once
 - it is extremely personnel-intensive, so the profit margin is small
- There will be a long-term move to a pyramidal model of consulting

The pyramidal consulting model

- In general, the 80/20 rule apply: “80% of the requests are easy, and can be answered immediately. The remaining 20% takes 80% of the total effort”
- So, you can take lots of customers, and get a small yearly consulting/support fee, and pay another company to answer the difficult questions
- For example: SAPdb (now MaxDB) consultants may have 100 customers, each paying 1k€ per year for support
- SAP offers 24/7 top-notch support from the same engineers that developed SAPdb at 30k€ per year
- ... so both are happy: consultants provide support for the 80% of requests and move the other questions to SAP, and SAP is paid for continued development

The pyramidal consulting model

- ... but it is necessary to create the structure, and the network of companies!
- This is a substantial problem right now: most of the FLOSS companies are very few and very small (in Italy there are approximately 150 FLOSS firms)
- Small companies can't survive by consulting alone, because it does not give a constant cash flow, so they do many different things...
- ...and this means that they can't focus on improving their skills...
- ...and so they are unable to grow.

Tools that may be of help:

- Flexible consortia
 - easy to create, easy to manage, limited effort
 - exclusivity or coordination should be explicitly spelled out in the beginning, to prevent conflicts among potential business opportunities
 - the usual approaches: subdivision by competence or subdivision by geographical area
- Non-vendor sponsored mediation services, for example “OSS help desks” or competence centers
- Better promotion of individual companies *outside* of the FLOSS community
- Help non-FLOSS companies understand the potential market of FLOSS services



Thanks for your attention

Carlo Daffara
cdaffara@conecta.it