

Contract no. 030776	Workpackage WP1	Delivery D1.5	Delivery Date 2007-08-31
------------------------	--------------------	------------------	-----------------------------

### 3 The Economics of F/OSS

F/OSS is a relatively new mode of software production. In this section we will briefly discuss its economic underpinnings. We will argue that there are compelling arguments for believing that the F/OSS mode of software production is sustainable and provides tangible economic benefits. We believe that such a discussion provides a necessary background for understanding the F/OSS business models. In the academic economic literature, F/OSS presents several puzzles. First, there is the *motivational puzzle*. Why do developers work on software whose value they cannot appropriate for themselves? The F/OSS licensing scheme makes software non-excludable and non-rivalrous in consumption. In effect that means that F/OSS can be considered a public good. This leads to a second question: *Why do private actors collaborate to create a public good, and how do they deal with free-riding?*

#### 3.1 The Motivational Puzzle

The motivational puzzle is: Why do F/OSS developers spend so much time and effort on a product that they essentially give away for free? Assuming a traditional economic perspective, this is only rational behavior if the benefits of doing this exceed the costs for the developer. The question then becomes, what are the benefits that individual developers gain? Lakhani and Wolf (2005) distinguish between two types of motivations: intrinsic and extrinsic motivations. A motivation is intrinsic when it is inseparable from the activity itself. Aspects of the activity itself motivate the developer. An extrinsic motivation is a separable consequence of the activity. Lakhani and Wolf subdivide intrinsic motivations into enjoyment-based and obligation/community-based motivations. As regards the extrinsic motivations, they take up the proposal by Lerner and Tirole (2002) that extrinsic benefits consist of immediate and delayed payoffs. Among the immediate benefits are:

- being paid by a company to work on F/OSS projects (e.g. Red Hat or IBM engineers working on Linux)

- being a direct user, and thus beneficiary, of the F/OSS that the developer works on

Among the delayed benefits are:

- career advancement (job market signalling)

- improvement in skills through collaboration and peer review

Lakhani and Wolf surveyed individual developers of F/OSS projects hosted on sourceforge.net. Their study started in 2001. The most important reasons that developers cite for their participation are that they need the software product they are working on (58%), and the intellectual stimulation provided by F/OSS development (45%). The third most important motivator is the opportunity to improve one's software development skills. The belief that software should be open is also an important community-based motivator (33%). In light of these results, Lakhani and Wolf conclude that "there is no single dominant explanation for an

individual software developer's decision to participate in and contribute to a F/OSS project" (p.19).

When the authors examine the factors that determine the level of effort the developers spend on F/OSS, they found that the only significant determinants are (in order of importance):

Enjoyment-related intrinsic motivations in the form of a sense of creativity

Extrinsic motivations in the form of payment

Obligation/community-related intrinsic motivations.

Empirical research on intrinsic motivations finds that in general extrinsic benefits such as being paid have a negative impact on intrinsic motivations. Lakhani and Wolf find, surprisingly, that this negative relationship is absent in F/OSS. For the large group of developers that are being paid for their participation (40% of the sample), the fact that they are being paid does not diminish the sense of creativity and/or community-involvement.

### **3.2 The private provision of public goods**

In the Lakhani and Wolf study of 2001 about 40% of the F/OSS developers were being paid for their contributions. Considering the increased involvement of companies such as Red Hat, IBM, and Novell in the F/OSS ecology since then, it is safe to assume that the relative size of this group has grown substantially in the last couple of years. This points us to our next question: *Why do private firms collaborate to create a public good, and how do they deal with the free-riding problem?*

Companies engage in F/OSS either by making developer resources available to a community-based project, or by setting up F/OSS projects themselves. The first case is exemplified by Red Hat that contributes heavily to the GCC (GNU Compiler Collection – the most important set of compilers for GNU/Linux). A good example of the latter case is Maemo. Maemo is a Linux-based F/OSS development platform for the Nokia tablet PC. Maemo was set up by Nokia with the express intent of promoting a developer community able to develop high-quality applications for the Nokia Tablet.

Companies are motivated to do this mainly for extrinsic motivations. Either they see an interesting business opportunity in F/OSS and adopt a F/OSS-based business model, or they do it for cost-efficiency reasons, or they do it for tactical and strategic reasons. The first group is exemplified by Red Hat or Novell that have adopted F/OSS based business models. These models will be discussed extensively in §5. The second group is typified by system integrators. System integrators like F/OSS because it enables them to increase profits through direct cost savings. The third group, those motivated to contribute by strategic or tactical reasons, is exemplified by the Maemo project discussed above.

Recent research suggests that some companies using F/OSS are also *intrinsically* motivated to devote resources to F/OSS (Rossi and Bonaccorsi, 2006). In this context, intrinsic motivation consists in adherence to F/OSS community values. These firms support F/OSS from a sense of obligation to give back to the community. As can be expected, Rossi and Bonaccorsi found that this holds for a respectable minority of firms (35%). The authors believe that these companies "have probably inherited their community-oriented attitudes from founders that were previously involved in [F/OSS] programming at the individual level and have turned their passion into a profession." (p.105).

Despite the existence of these intrinsically motivated companies, it is clear that most companies engage in F/OSS for opportunistic and self-centred reasons. Many observers fear that these companies will "take out more than they put in". More precisely, they fear that companies will try to appropriate as much value as possible from a F/OSS project (e.g. by offering complementary services), while committing as few resources as possible to the

project. This is a case of the “free-rider problem”. Here is how Wikipedia describes it: “free riders are actors who consume more than their fair share of a resource, or shoulder less than a fair share of the costs of its production. The free rider problem is the question of how to prevent free riding from taking place, or at least limit its negative effects. (source: [http://en.wikipedia.org/wiki/Free\\_rider\\_problem](http://en.wikipedia.org/wiki/Free_rider_problem)).

In general, free-riding behavior is seen as destructive because it tends to undermine the motivation of those actors that do take up their fair share of the burden. Recent research, however, suggests that F/OSS is relatively insensitive to free-riding behavior. Rossi and Bonaccorsi, in a study on the contributions made by firms to F/OSS find that “open Source communities permit some member to take much more than they give, provided they do not violate minimal membership rules. ... The literature on [Common Property Regimes], public good provisioning and free riding has probably overestimated the potential destructive role of a small number of non contributors, assuming that their behaviour should inevitably self-propagate. This is not necessarily true” (Rossi and Bonaccorsi, 2004).

We believe that the relative insensitivity of F/OSS to free-riding is the consequence of several features of F/OSS. The first, and perhaps most important is that F/OSS presents to many companies a third option to the traditional “make-or-buy” dilemma (Bessen 2006). Companies that have highly specific needs, can collaborate with other companies and individual developers on a “platform” that can then easily be customized for their own specific purposes. A very prominent example of this is Linux. Linux is used as operating system by mobile phone (Motorola), consumer electronics (TiVO), and computer equipment vendors (IBM). The modularity of Linux makes this broad array of uses possible. It provides a powerful and stable “platform” that serves as a point of departure for the custom developments that these companies require. Because the collaborating companies benefit from a high-quality base product, they are motivated to co-operate to ensure the continued existence and quality of the base product. Free-riding is not an option for these companies because they have highly specialised needs: no other company or set of individuals is likely to produce exactly the product that they need. Baldwin and Clark (2003) also argue that free-riding in F/OSS projects will be limited if the product is highly modular. By “modularity” the authors mean that individual features can be added to the software base product easily, incrementally and independently from other project participants. Highly modular software with a very large potential feature set lowers the cost to develop an additional feature, while making free-riding behavior unattractive: because a relatively small group of developers can select features to implement from a large set of potential features, free-riders face a low probability that a developers implements the feature the free-rider wants.

The argument that modularity limits the damage from free-riding is strengthened by two other aspects of software in general and F/OSS in particular. One is that “platform” or “infrastructure” code is actually not necessarily very resource-intensive. For instance, in an audio interview, Mark Shuttleworth, the multi-millionaire founder of the popular Ubuntu Linux distribution, commented that he could keep the Ubuntu project funded for “a couple of decades if necessary” out of his own pocket. Something similar holds for the Apache webserver. The study of Mockus, Fielding and Herbsleb on the Apache project found that a mere 15 core developers contributed more than 80% of the source code (2000). These relatively small teams are able to keep the platform in good shape so that others can contribute small, incremental features that collectively increase the value of the total product. This also suggests that the cost of contributing on the platform is for many companies sufficiently small relative to the value of the complete product that most companies will not free-ride: the potential bad press and bad community relations are simply not worth the cost-saving of free-riding.

The second feature that makes F/OSS resistant to the damage of free-riding is that software in general is characterized by strong positive network effects. The value of a piece of software increases as the number of users of the software increases. In the case of F/OSS, companies that free-ride are still increasing the uptake and usage of the software. The increase in popularity of the product will usually imply an increased motivation for the individual developers (for intrinsic reasons such as pride in the result, but also extrinsic reasons,

signalling one's programming abilities to a wider audience, e.g.). So the negative effects of free-riding are dampened by positive network effects.

In case the F/OSS is covered by a reciprocal license such as the GPL, the licence itself lessens the damage of free-riding. Although free-riders profit from the fruit of someone else's work, they cannot take it out of the commons and turn it into a commercial product. So free-riders cannot do very much damage because they cannot create a rival product to the original.

A final reason why free-riding is not so damaging in F/OSS, is that knowledge is an intangible asset. The companies and individual developers that contribute to F/OSS has a depth of knowledge about the product that free-riders have not. Although, in principle everything is open to view and available to anyone, in practice in-depth knowledge can only be acquired by working on the code and start developing in it. In fact, customers of F/OSS services know this also, and therefore will prefer to acquire services from a company that has made genuine contributions to the F/OSS rather than from one of its free-riding competitors.

This is not to say that free-riding provides no problems for F/OSS. Significant free-riding can diminish the chances of success of a F/OSS project. But it does seem that free-riding is less of a problem than in other domains.