Free Software Business Models

Jim Kingdon
Cyclic Software

Abstract

Free software, in addition to being an activity for users and volunteers to participate in, is a viable business model. Free software businesses have learned various things about how to make it work.

Five successful ways of making money are by providing custom work for ports and new features, support contracts, training, consulting/customization, and telephone support. Documentation and internal code cleanups are probably better treated as overhead rather than being funded specifically. Funding major projects can be difficult—various models have succeeded in the past but it isn't as straightforward as selling support for existing software.

It is beneficial to collaborate with developers outside the business, who are working on the same software. Techniques include a mailing list focused on improvements people are planning to make, and offering checkin access to people who have demonstrated their ability to contribute. Widespread distribution of the software can publicize the software and help people conduct pre-purchase evaluations. Two ways to enable people to redistribute the software and offer support are (1) have the redistributor sell support contracts, and have an agreement which provides for passing along difficult problems for money, and (2) have the redistributor provide only the most minimal support themselves, and refer people if they want a greater level of support.

Using these free software business models, there is no limit. Free software can reach software intended for hackers and software intended for non-technical people. People in free software businesses can make a living. And free software businesses need not inhibit in any way the communities of users and contributors which have been so instrumental in many free software projects.

1 Introduction

Freely redistributable software, which I will refer to as free software for brevity, may be most familiar to many people as an activity driven by users and volunteers. Users may write improvements to the software, or assist in its maintenance so that they (or their employer) can use it. Students or others seeking to develop their skills may work on the software as an educational project. Researchers may implement improvements as a way of trying out new technical ideas. And of course, anyone may contribute for the sheer fun of hacking. All of these types of contributions are not to be discounted, and in fact are very valuable to the progress of free software. However, it is a misconception to think that free software (particularly the best and most widespread free software) has relied exclusively on mechanisms such as this.

For instance, X Windows has been written and maintained by a consortium, funded by the companies which use X Windows in their products [XcoWba], which employs some 25 people [XcoWbb]. The Free Software Foundation has written and/or maintained GCC, GDB, emacs, and other GNU programs. They are funded by donations and their distribution services [GNU96a,GNUWba]. They employ some 8 people [GNU96b]. Cygnus Support has been responsible for much of the recent development of programs such as GCC, GDB, and binutils; they employ over 50 people [CygWba]. This is only a sampling; there are others as well. This paper does not try to list every possible funding mechanism, but it does describe some of the basic ways that one can make money off free software.

2 What people will pay for

In years of selling free software support, people in the business have developed some experience about what works and does not work. Here are some of the services which people have been willing to pay for.

A prospective customer may approach the free software business with an inquiry such as: "We like the software, but can you make it do X?" Examples are porting the software to a new platform or adding a particular feature. Some free software businesses will respond to such an inquiry by quoting a price for adding the desired functionality. The parties will typically sign a contract detailing the work to be done, the timetable, and a price, generally fixed, to be paid for the work. Two companies offering this kind of service are Cyclic [CycWba] and Cygnus [CygWbb].

Another successful model is to sell yearly support contracts. They are similar to the support contracts sold by many vendors for non-free mainframe and unix software, and provides fairly comprehensive service—in particular both assistance using the software (typically by telephone or email) and fixing bugs. By having a fixed yearly price rather than charging for each problem resolution, the free software business has an incentive to maintain a quality product (in those areas that customers care about). However, many users find that such contracts are too expensive. They don't expect to need a lot of bug-fixing and assistance. For example, they might be willing to work around bugs rather than get them fixed. Two companies offering this kind of service are Cyclic [CycWba] and Cygnus [CygWbc].

A support offering which may be more appropriate for some customers is telephone support. This is priced per-minute or per-call, allowing people to use the service with no big financial commitment. This service is often most useful for people seeking assistance but also could encompass bug-fixing, at least for some kinds of bugs. One company offering this service is Yggdrasil [YggWba]. It is probably not as profitable as the more expensive yearly support contracts [Per95a], but seems to be viable.

Training classes are a business which is well-established at least for non-free software. Classes may be located at the customer's site or the training company's site. Classes typically have a standard curriculum. An example of a company offering training (for non-free software) is The Learning Tree [LrnWba].

Another well established business, although not necessarily in a specifically free software context, is hourly consulting or customization. The customer will have the consultant come in, look at the customer's use of the software, and offer advice, help set up the software in a way that matches the customer's needs, or write software to meet a specific need. The most common arrangement is that the customer receives ownership of any software produced. Providing for the software to become free software instead would change the business arrangement relatively little in many cases.

3 What people (usually) will not pay for

There are other activities that one might hope to find a customer to fund, but have proven to be difficult to do that way.

One is internal code cleanups. One can include some cleanups in preparing quotes for ports and new features, but you won't tend to find customers willing to pay $40,000 to rewrite the foo, bar, and baz modules when the alternative is paying $10,000 for a less clean solution, doing without the feature, hiring someone else to implement it, switching to a competing product, etc.

Another is documentation. People rarely approach free software businesses asking to fund writing better documentation. The one exception that I am aware of is that the Free Software Foundation did receive a donation specifically earmarked for writing the *Emacs Lisp Reference Manual* [Per91a]. Also, customers who are paying for adding new features will pay for documentation of those features [Per96c].

The fact that it is hard to get customers to fund these items specifically does not mean that they cannot get done; it merely means that one can consider them as overhead, funded by services such as support contracts and training classes.

4 Funding major projects

So, if one of the standard ways to enhance the capability of a free program is to find a customer who wants to fund the work, what does one do if the work is too large, relative to the benefit for any one customer? Experience has shown that graphical user interfaces are often in this category.

One approach is, of course, to simply fund the work oneself. This works as long as the improvement will make the software sufficiently more attractive that one can sell enough additional support contracts to make back the investment. For example, Cygnus did its Graphical User Interface for the GNU Debugger this way [Cyg95a].

Another method is to line up a bunch of funders each of whom is willing to chip in a small amount. For example, Cygnus sought such customers for the initial Solaris 2.x port of the GNU software development tools [GNU92a]. This is a very logical approach—often in such situations people will be not only willing but eager to participate. But such projects can be hard to manage. The difficulty is lining up a sufficient amount of funding from people who are all interested in the same thing (not just generally similar things). Timing is also critical; by the time the last people show up, the first people (or their employers) may have lost interest.

Another way for major projects to happen is outside the context of a free software business. Most free software businesses have been built upon existing software (perhaps arising from a research lab, a different business, or a volunteer effort), because of the difficulty of funding major projects.

5 Developers

Because the software is free, diverse people will be enhancing it. A free software business needs to figure out how it will relate to other people working on the same software but not employed by the business (who I will call *developers*). One possible answer, of course, is to let them go their own way. But there are various business reasons to involve them. One is that they will contribute bugfixes, features, and ideas that customers want. That way, when a customer request comes in, a free software business can simply inform them about the feature or perhaps tidy it up a bit as needed, rather than developing it from scratch. Another reason to cooperate is that, without coordination, developers will tend to create their own branches of the software and then customers will complain that the branch they are getting lacks features from the other branches.

On the other hand, if not done carefully, interacting with the many, many people out there who might be working on the software or attempting to do so can be a huge time sink. There are proven techniques to help with this.

One is offering checkin access, that is, the ability to make changes to the master sources. Many free software projects have been held back by the situation of an overworked maintainer trying to keep up with reviewing all changes and putting them in. One good solution is to recruit other people who are qualified to determine what is a good change. Not everyone who volunteers is qualified, however. By requiring people to submit at least one, and preferably several, good quality changes before being offered checkin access, one can weed out people who never get around to writing any code and people who write poor quality code.

Of course, when there are multiple people with checkin access, they need to communicate regarding criteria for checkin. So it is a good idea to have a mailing list for those with checkin access. Because this list is so central to the functioning of the developer group, it is important to keep it focused on concrete improvements that someone has offered to implement, not on pipe dreams. If this ground rule is established, asking "who has volunteered to implement this?" can be very effective in curtailing a discussion which is getting off track.

Free software projects using the mechanism outlined here, or some variation thereof, include CVS [Cyc96a], FreeBSD [FrBWba], and NetBSD [NtBWba].

6 Wide distribution of software

Because the software is free, various people will, of course, distribute it. So the question for the free software business is how to participate in this. Should a business make binaries of the software available to anyone on the internet? Or should it just provide them to customers (and figure that others can take care of further distribution if they choose)? To what extent should a free software business promote the software's availability to noncustomers?

There are various advantages to the free software business to having everything—sources, binaries, and printed manuals—widely available. One of the more obvious, for some entities, is that they run distribution services which bring in needed revenue. Examples are Red Hat [RdHWba] and the Free Software Foundation [GNUWba]).

Another advantage is that, particularly if the product consists of relatively expensive support contracts, many customers will want to evaluate the software before buying. Vendors of non-free software often have demo versions or limited time licenses. For the free software business the most obvious answer can also be quite effective: "just download it!"

Wide distribution also promotes word of mouth marketing. The more people who are using the software, even if they are not people who are interested in the services of the free software business at the time, the more likely that it will spread to people who will use those services. For example, a number of our customers at Cyclic had been using CVS for some years before they purchased support.

Now, this fact—that being associated with the software which is widely available can lead to greater exposure—also has a downside. If someone has a bad experience rather than a good experience, then that is also more likely to reflect on the free software business that has an association with the software. It is probably impractical to provide sufficient documentation, mailing lists, and other mechanisms so that *everyone* who tries the software gets the same kind of help (and therefore as favorable an impression) as a customer would, so it is useful to make it clear that support is an extra service which one cannot expect if one merely gets the software alone.

7 Redistribution of support

One of the key possibilities facilitated by free software is that it becomes possible to get the software and be able to redistribute it. For example, a hardware vendor may distribute the software with their hardware. In the context of a free software business, though, one must ask about how one handles support for the redistributed software.

It is impractical to support a customer and all of their customers for the same price for which one would support only the customer themself. Redistributor arrangements, in which people who receive the software have ways of getting support if they need it, are essential to getting into as many markets as possible. Non-free software tends to be sold through a wide variety of resellers.

One way of solving this problem is for the redistributor to sell support contracts, with an arrangement by which easy problems can be handled directly and harder problems can be passed along, with the revenues divided between the companies in some way. For example, SGI has an arrangement with Ada Core Technologies to redistribute the GNU Ada compiler in this fashion [ACTWba], [Per96a].

Another solution is for the redistributor to provide minimal or no support (for example, installation support only), and then suggest that people buy a full support contract from the upstream company directly if they want a greater level of support. Hitachi distributes compiler tools from Cygnus on this basis [HitWba], [Per96b] and at one time Wind River also distributed those tools similarly [Cyg95b].

8 Conclusion

The various success stories mentioned above should make it clear that one can make money off free software. Also, communities of developers and users (taking a more active role than merely as customers) and free software businesses complement, rather than undermine, each other.

By looking at the particular models and the ways that they provide revenue, an even more sweeping conclusion emerges. The business models described still work even if the vast majority of users do not have the skills or inclination to modify the software. Applying these models shows how to make free software work for most or all kinds of software. Software for hackers, the kind of free software which has been most successful to date, is only the beginning.

References

[ACTWba] "A Well-Established Product," section of "GNAT maintenance," sub-page of the Ada Core Technologies web page at `http://www.gnat.com/`.

[CycWba] "Prices for our CVS support," sub-page of `http://www.cyclic.com`.

[Cyc96a] "Cyclic Thanks CVS Contributors," news release of 26 Aug 1996, available from http://www.cyclic.com/.

[Cyg95a] "Cygnus Support Ships Graphical User Interface for GNU Debugger," press release of April 17, 1995, available from http://www.cygnus.com/.

[Cyg95b] "Wind River and Cygnus Support Announce Partnership," Cygnus press release of September 11, 1995, available from http://www.cygnus.com/.

[CygWba] http://www.cygnus.com/.

[CygWbb] "Cygnus Products," sub-page of http://www.cygnus.com/.

[CygWbc] "Cygnus GNU Professional," sub-page of [CygWbb].

[FrBWba] "FreeBSD contributor list," section 26 of the *FreeBSD Handbook*, available from the "Documentation" sub-page of http://www.freebsd.org/.

[GNU92a] "A GNU Standard on Suns?," *GNU's Bulletin*, Free Software Foundation, Boston, MA, Vol. 1, No. 13, June, 1992. Online at http://www.cs.pdx.edu/˜trent/gnu/.

[GNU96a] "What is the FSF?," *GNU's Bulletin*, Free Software Foundation, Boston, MA, January, 1996. Online at http://www.gnu.ai.mit.edu/.

[GNU96b] "GNU's Who," *GNU's Bulletin*, Free Software Foundation, Boston, MA, January, 1996. Online at http://www.gnu.ai.mit.edu/.

[GNUWba] Free Software Foundation, Boston, MA, http://www.gnu.ai.mit.edu/.

[HitWba] "Hitachi SH-1 $99 Eval Board," sub-page of http://www.halsp.hitachi.com/ as of December 1996.

[LrnWba] Learning Tree International, http://www.learningtree.com/.

[NtBWba] "The people who make it all possible," sub-page of http://www.netbsd.org/.

[Per91a] personal communication from Robert J. Chassell, about 1991.

[Per95a] personal communication from someone who worked for one of the companies to whom Microsoft outsources Windows 95 support, 1995.

[Per96a] personal communication from sources at Silicon Graphics Inc (SGI) and Ada Core Technologies, May 1996.

[Per96b] personal communication from Stan Shebs of Cygnus, December 1996.

[Per96c] personal communication from Roland Pesch, December 1996.

[RdHWba] http://www.redhat.com/.

[XcoWba] "What is the X Consortium," sub-page of http://www.x.org/ as of December, 1996.

[XcoWbb] "The Staff Here at the X Consortium," sub-page of "Job Listings and Staff," sub-page of http://www.x.org/ as of December, 1996.

[YggWba] "Technical Support Policy," sub-page of "Yggdrasil Technical Support Information," sub-page of http://www.yggdrasil.com/.

Biography

Jim Kingdon has worked in a number of free software businesses, including Cygnus and the Free Software Foundation. He is currently with a small free software startup, Cyclic Software, and as such does some of just about everything, including management, sales, and marketing.