The Linux Business Model

Nils Löhner
lohner@spi-inc.org

June 17, 1999

Contents

1 Abstract

The Linux world is a new and different place for people used to the computer business in the traditional sense. We will cover the basics of how the Linux world is set up and how it differs from the current development and business models, and examine each piece one at a time and analyze it from the Linux perspective. Discussion will focus on software and hardware development, sales, and support.

Software development under Linux has changed the way software is being developed and viewed in the marketplace. There are thousands of programmers all over the world volunteering their time and effort to help write the software for the Linux kernel and its surrounding operating system. This development is occurring on diverse platforms from x86 to PowerPC, from Alpha to UltraSPARC. These programmers are the basis of the Linux business model, and are the single most valuable resource in the community. They are the reason the business model must be structured around them. Trying to fit them into another business model can be very difficult and will likely prove counterproductive.

As there now exists an operating system that will run on an extremely wide range of hardware, software developers can now write one peice of software and have it run on any number of different platforms without requiring a rewrite of the software. This enables them to reach a wider customer base. The customer base also has more of a choice in hardware- if the same operating system runs on a wide range of hardware, then the customer can make a choice based more on price/performance instead of a compatibility based choice.

Organizations can profit from these changes in the development model, but will need to change their current way of thinking as the old set of priorities that decisions were based on have now changed.

As the topic that this paper is attempting to address is not stagnant but rather continually changing and evolving, this paper will gradually be revised to reflect the state of the industry with respect to the material covered. This paper does not attempt to provide answers to all of the questions that are raised, but rather provides thought provoking material that will give the reader an insight into the Linux development world and provide some ideas on how to interact with it.

2 Overview of Background Material

This following sections cover some basic material on which much of the remainder of this paper is based. A basic familiarity with the computer industry and its workings (both development and marketing) is assumed.

2.1 The Computer Development Tree

Figure 1 shows the basic computer development tree which, at this level, does not appear to be anything out of the ordinary. The differences between this model and others are found at more microscopic levels which will be discussed in more detail throughout this paper.

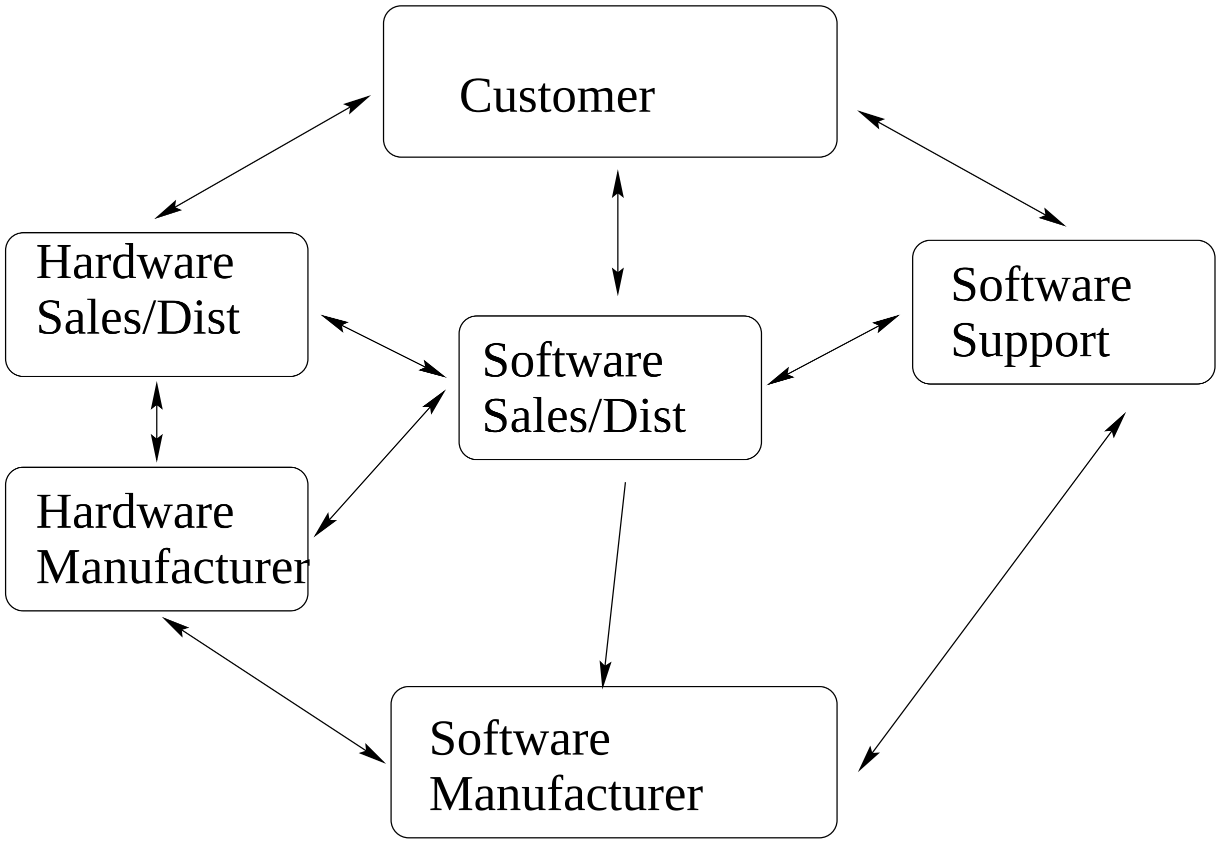

Figure 1: The Computer Development Tree

2.2 The Computer Product Evolution Stages

The evolutionary stages of computer products play an important role in the development of products, and thus it is important to take a brief look at the different steps present in this evolution.

- initial concept

  The developers arrive at the initial idea for the product.

- initial implementation

  The product idea becomes a product that can be tested on the market to see if it meets the needs it intends to fill.

- initial usage

  The product is used and the market decides whether or not it is useful.

- industry acceptance

  The industry decides that this product is useful and begins using it and integrating it into day to day operations.

- further development

  Other manufacturers and developers pick up the idea and try to improve on it and try to win market share with their product through competitive differentiation (a superior feature set, better performance, etc.).

- market saturation

  Many similar products appear on the market, and the consumers will decide which are and which are not successful.

- product refinement

  The products learn from the features of other products and evolve to contain a superset of all useful features for that product.

- standardization

  The market decides what features are needed for the product to accomplish its task and begins to favor the products that have the most suitable ('standard') feature set.

- 'survival of the fittest'

  The market has decided what features constitute a successful product of that kind and products that do not have the required features begin to disappear from the market. At this point it is also not worth entering the market unless on can substantially improve on a product already in existence.

Some good examples of this are word processing and spreadsheet programs. Initially, after a need for these types of programs was demonstrated, there were many programs competing in the market. They had many different interfaces and features, but gradually they developed into word processors and spreadsheets as we know them today: with a fairly uniform set of standard operations and similar user interfaces. Examples of this evolution can be found for hardware as well as software.

It is important to keep this cycle in mind when entering the computer development arena, as the marketing and development strategies (and the product goals) are different at each step of the process.

3 Software Development

The open development of software is the keystone of the increasing success of Linux. With many programmers freely contributing their time and code to the system, it has unprecedented growth and also very rapid technological advancement for a mainstream operating system. In order to understand the basis of this business model, several areas of this new software development methodology must be examined in detail.

3.1 Development Models

Software development can be split into three major areas. The developers as well as the goals of the development in each area are different in nature.

3.1.1 Open Development

This type of development is done by people whose main interest is to produce a working piece of software that satisfies a given need. This leads to a piece of software that is functionally very capable, but does not have all of the necessary features (well designed interface, sufficient functionality, user documentation, etc.) to satisfy a market need. The features of the software (or lack therof) will determine the success of the software in the 'industry acceptance' stage of the evolution of the software.

Open developers are in a completely different class from any other set of developers. Their main objective is to put out working software to accomplish a given task. The work here is done mostly on a volunteer basis, and the product is available to any consumer free of charge, source code and all.

The reasons for writing software that is freely useable, distributable and modifyable include personal pride, wanting to give something back to the community that produced code which they have used for so long, and peer recognition (the 'kudos factor').

The advantages of this model include very rapid development of software as the developer base of the software can grow with the program. There can also be a large influx of new ideas into the product that can quickly catapult it to the leading edge of technology.

3.1.2 Commercial Development

This is the 'standard' development method for software. A company decides to produce software to fill a given market need, and pays its employees to design and develop the software.

It should also be noted that there is a difference between proprietary software and commercial software. Commercial software can be thought of as a well known set of functions that has been implemented by a company. Proprietary software includes some technology that is not disclosed to the public, thus preventing other developers from leveraging the technology for other uses. It does not help the community as a whole as the technology can not be leveraged for other products. This may lead to less widespread acceptance, and if the functionality of the software is desirable, a free alternative will likely become available at some point.

3.1.3 Open Development with Commercial Investment

The key to successfully selling software in this new environment is providing added value such as an improved interface or documentation. Many organizations are turning to support as a means of generating revenue from openly developed products. If an openly developed product manages to work its way into the market, then the need for support will follow. Companies may also choose to develop additional features for the software at the request of their customers.

3.2 Software Licenses

Free software is distributed under a variety of licenses. Commonly used licenses include the GNU Public License (GPL), BSD, and Artistic licenses. This paper will not go into details on these licenses, but more information may be found by searching the Web.

4 Hardware Manufacturers

Hardware manufacturers are faced with interesting choices in this development model. They can design their hardware and write all of the necessary drivers for any platform themselves, or, after the design stage, they can release as many specs about the device as possible, and enlist the support of the open software community in writing the drivers. This allows for the most rapid progress, as the designers of the devices along with their software support people can now simply organize the writing of the driver software and ensure that it works as opposed to having to design, write, test (debug, release, debug, release, debug...), and release it themselves.

Hardware vendors who will give out the specs to their hardware for the drivers to be written and even help with the efforts will get a wider platform usage (and thereby possibly market acceptance) from their hardware. For instance, if a board is designed for the PCI bus, and the specifications are available and the code is open to the public, it will soon be ported to other systems using the PCI bus. This increases the vendors customer base.

Hardware manufacturers also have choices to make with respect to other system specific software. Most hardware vendors have written compilers, low level graphics drivers and other software specifically for their platforms and devices. This software may or may not contain proprietary information. The question is whether this should be rewritten, ported, modified and released, etc.

Releasing the code for this software will enable the community to latch on to that software and leverage the technology present in it. The manufacturer gains a workforce that is willing to invest in the development of the software because the hardware product is thought to be worthwhile. By not releasing software a manufacturer may be protecting proprietary information, or parts of the software that the manufacturer considers to be critical factors in the competitive differentiation of their product.

Let us briefly examine the options available to hardware manufacturers and the advantages and disadvantages of each option.

4.1 Rewriting Software

Some pieces of proprietary software may need to be rewritten (in order avoid releasing proprietary information) to release them to the general public. The manufacturer still runs the risk of revealing architectural or other information about the hardware to competitors, but now the development of the software can happen openly, which can lead to porting to other platforms or perhaps other interesting technological improvements.

4.2 Porting Software

Much of the software that is architecturally specific (compilers, graphics drivers, etc.) currently has a duplicate copy floating around in the free development world. There are several interesting points here that should be made.

- anything written architecturally specific will run better that a generic piece of software

  This is due to the fact that the HW vendor can optimize the software specifically for his hardware, using it to the limit of its capabilities.

- anything written that is available freely is not (and can not be) architecturally optimized for every architecture, but it is generic and will run on any platform

  This enables more widespread use of the software but inhibits performance.

Porting from the architecturally specific code to something that is usable on any platform is an incredibly big amount of work. The hardware vendor may choose to go an easier route and simply port the optimized back end (the portion that is designed for his specific hardware) to an already existing standard front end. This approach takes the best of both worlds, as there is less work to do for the manufacturer, and the customers of the HW vendor can still enjoy the enhanced performance of the optimized software.

5 Hardware Retailers

Hardware sales also become interesting at this point. The retailer has to decide which distributions of Linux to include with their system, as well as tailoring a set of application software to the customers needs. It is probably prudent to consider a partnership with a software retailer for this purpose, unless a large enough technical staff can be assembled. Remember, support will also be requested by the customer. The customer wants to buy a solution from the vendor, and not just a piece of hardware that he must invest countless hours in in order to install, configure, and maintain the system. There are exceptions to this such as ISPs, scientific institutions, etc. These organizations may need such a customized setup of their systems that they may simply invest in the personell to administer the systems themselves.

5.1 The Hardware Sales Picture

With Linux able to run on such a wide variety of hardware, several new possibilities become viable. If the same application software can run on a wide variety of machines, then the customer can buy the hardware with the right amount of power for ther user, and save money by buying less powerful (even different!) hardware for different users requiring different performances from their machines.

The sale of hardware is no longer dependent on the set of application software available for it, but rather on meeting the specific performance needs of the customer. The customer can now run the same operating system throughout the company to ensure compatibility among all systems, and can easily change the performance point of a system (i.e. upgrade a slow NFS server) simply by changing one piece of hardware- a piece of hardware that is no longer vendor specific.

5.2 The Business of Used Computers

There is now a new way to make money in the industry. Used computers can be used in conjunction with operating systems such as Linux to be web clients, web and FTP servers, research engines, routers, firewalls, and many other tasks that are not necessarily computationally intensive.

The markets for such hardware include developing nations, schools, libraries, and possibly even research facilitles where they can be networked into a more powerful system throuch software that allows distributed processing and resource sharing. All of a sudden a computer that was collecting dust has become a valuable resource to someone (and a valuable market to someone else- the retailer).

6 Conclusion

The computer market (both hardware and software) has been changed irrevocably by the emergence of Linux and the development models around it. This is still very much a developing industry, and there are many opportunities for business, but one needs to look closely at the inner workings of the industry in order to be successful.

When considering the computer evolution stages, it becomes apparent that at the end of the evolutionary cycle there will be less products of a given kind on the market, but their use will be more widespread. At that point the demand for support and service will be reasonably high (and sustained as opposed to the one time cost of sale) thus enabling a vendor who has expertise in administering a given product to continue to work in the market of that product through support.

Given the information presented in this paper, the bottom line for success in this business model appears to be that one needs to add value to an existing product in some way to achieve success. This holds for both software and hardware. For software, increased documentation or user friendlyness are greatly desired, and for hardware the specifications that will enable developers to easily write and port the drivers to a variety of platforms can lead to success.