Synopsis 1: Open Source
CompSci 82 (Astrachan)
[redacted]

In "IBM's Pragmatic Embrace of Open Source," Pamela Samuelson explains how, over the past 20 years, IBM has shifted from vigorous support of intellectual property rights for computer programs to embracing open source software while still developing and licensing some proprietary software. IBM's original business model emphasized restricting access to source code and patenting and licensing their products. Their approach to software was the complete opposite of General Public License developer Richard Stallman who argues that software code should be made available to the public so that adaptations can be made and shared with others.

Samuelson proposes three reasons for IBM's adoption of open source. First she describes how, during the 1980's, Microsoft and IBM partnered to develop a new operating system, OS/2. At the same time, Microsoft was developing a competing operating system, Windows 3.0. After the overwhelming success of Windows, Microsoft moved away from the OS/2 model, ceasing their working relationship with IBM. In the end, the IBM's OS/2 model was not successful in the marketplace, largely because it was incompatible with Windows. Samuelson suggests that IBM's adoption of the open source operating system Linux may be due, in part, to the fact that Linux poses a threat to Microsoft's essential monopoly on the operating system market.

Samuelson also suggests that over the years, IBM's business model changed once Louis Gerstner became its CEO in the 1990's. Gerstner believed that IBM should become more customer-focused. Linux is a product that supports customer needs because it can be tailored to them. IBM profits from open source, particularly Linux, because it is less expensive to develop and it is a jumping off point for designing special services and applications that run on top of Linux.

Finally, IBM has embraced open innovation. A key advantage of open source is that the costs of designing, developing and improving software are spread out among various contributors; expenses are lowered when the effort is shared. Also, when IBM adopted Linux it already had a substantial customer base so they did not incur any of the typical launching costs. Collaborative development has made software more interchangeable and has led to increased innovation in products and services. In conclusion, Samuelson explains how IBM's shift toward open source is representative of changes in the nature of the software industry in general over the past 20 years.

After reading this first article and skimming the two Wikipedia entries I was left with some questions. My first question was, if open source software is just as effective, if not better, than proprietary software and it's free, how can firms relying on proprietary software survive? Is it more likely that firms will continue to mix open source and proprietary software projects rather than shifting to a solely open source model? Also, how can programmers make a living if they are not receiving compensation for the enhancements they are providing? How do they receive recognition? Is it fair to say that the open source pool is more talented given that it is so much larger than the pool from a single firm? Many of these questions were answered in the second article I read.

In "The Economics of Technology Sharing: Open Source and Beyond," Josh Lerner and Jean Tirole attempt to explain the recent rise in corporate investments in open source projects from an economic perspective. While the programmer incurs an opportunity cost of time, there are both short-term and long-term benefits. Open source programmers may later improve their performance in paid work and have future job offers, and they may find more intrinsic enjoyment in working on open source projects, and they may find ego gratification from peer recognition.

Empirical evidence suggests that individual contributors benefit directly, even though proprietary programmers make better compensation.

Lerner and Tirole address issues related to commercial firms and open source, suggesting that some firms "live symbiotically" by supplying "services and products that are complementary to the open source product" (105). They also touch on some legal issues related to licensing. However, I found their discussion on the relative quality of open source particularly interesting. Lerner and Tirole argue that bugs are fixed significantly faster in open source projects. Open source projects also can be tailored to particular customers' needs. Other authors suggest that, since open source programmers are not paid for their work, they will not be tempted to engage in particular types of collusion. Also, open source supporters argue that, because programmers can identify bugs more easily, open source software is more secure. However, proponents of proprietary software say that open source software is more susceptible to malicious hackers. According to Lerner and Tirole, public policies supporting open source tend to have an unclear effect on social welfare. From one perspective, open source allows any user to access particular software, but from another perspective, developers may stop introducing new products due to a lack of incentives. Finally, Lerner and Tirole suggest that there are additional benefits for corporations switching to open source. They will not be limited by future price increases because they can always make adjustments to existing technology to suit their needs. They will avoid the "patent thicket". And they will help establish a particular technological standard.

In addition to explaining open source from an economic perspective, Lerner and Tirole explain it from an academic perspective. Academics do not generally receive direct financial return for their work, but they still enjoy the peer recognition. They also select topics based on the fields where their contributions will be most beneficial; the same goes for open source

programmers. A key difference, however, is that while academic work is not generally well-disseminated, open source code is made widely available to the public.

In general, I found this second article to be very helpful. It was clearly written, used straightforward, concrete examples, and it relied upon analogies that were more accessible. It answered many of the questions that I had generated after having read only the first article and the Wikipedia pages.