

# How to foster innovation

Eric Blair

4 February 2005

**It takes a harem** Most programmers are male, and socially inept. Like most males, they like sex with cute people. This has a simple policy implication: if we want to encourage innovation in computing, we need to reward programmers with harems. Programmers would submit their programs to a review board, perhaps at the USPTO, which will evaluate the proposal's merit. Those with low merit will earn head points; those with exceptional merit will earn a nineteen year old love slave—or two—as the inventor's sole property. The love slave him/herself may be unhappy with the deal, especially given the typical programmer's hygiene habits, but it is essential that we foster innovation, so the overall program is certainly worth it.

Every time somebody proposes stronger patent laws, they defend the statement by explaining that we need to foster innovation. They take this as sort of the discussion-ending argument—how can you be opposed to innovation? But there's a cost-benefit analysis that needs to be done. A patent is a monopoly (OK, a limited right to exclude) and that means that those who would have been competing, and those consumers who would have had multiple products to choose from, could be worse off. This is especially true if the innovation would have happened anyway; see below.

There are people who advocate against doing a cost-benefit analysis. Here is the American Intellectual Property Law Association's response to recommendations about reforming patents by the FTC. For those of you who don't feel like reading white papers by lawyers, here's the gist: the AIPLA advocates all of the recommendations by the FTC which would expand the need for more patent lawyers, such as a post-grant review process, and opposes those which would diminish the scope of patents, such as a cost-benefit analysis of whether patents in a given industry would actually help the economy along. [In particular, see comments on recommendation #6. They claim that the courts can not do a cost-benefit analysis—true—and that the Congress, which is constitutionally obligated to do the cost-benefit analysis, has already considered whether software is patentable—false. The last law written by Congress which covers software patenting was written in 1952. Not thinking about it is not equivalent to reasoned consideration.]

People will tell you that we need to foster innovation, and you can respond: at what cost? Should we sacrifice some virgins to the mysterious gods of innovation? Should we believe Microsoft when it tells us that it just can't afford to

do innovative research unless it has an absolute lock on the market?

**The transmogrifier** Every science fiction story has a device which will make anything. You walk up to it and say, ‘I’d like a soylent blue patè and a plate of wheat crackers, please’, and it would suck in ambient particles and produce a plate exactly to your specifications.

Under current patent law, if you can demonstrate that you are the first to make this request to the transmogrifier, then you can file for a patent: “A method and apparatus for producing soylent blue patè with accompanying bread product.” The work wasn’t much: you just had to ask. But since you were the first to do it, you are the innovator, and can exclude others from using the transmogrifier to produce soylent blue pate with crackers.

The computer is a data transmogrifier. If I need a program to index soylent blue recipes, all I have to do is ask in the right language. If I need a program to calculate maximum likelihood estimates of the parameter in a Zipf distribution, I can write one. [I actually have such a device in the other window.] It’s not as simple as asking nicely, but if I give a competent C programmer a flowchart for a program I want, I know with 100% certainty that he or she will give me something that basically meets the specs.

There are loads of details which need to be worked out, and no two programs will look exactly alike, but translating an idea into computer-readable language is a different sort of inventiveness from what we’re used to in the physical world. Let’s say that I’d like a solar-powered machine that can mine slate quarries and bake delicious cakes. If I hire a competent mechanic, he or she would need to spend time in designing and building physical objects which might work on paper but probably won’t work the first time they’re built. For the computer, all I had to do was write a flow chart, and then the implementation is hard but guaranteed to work. For the physical machine, the flowchart wasn’t enough: just because it looks good on paper doesn’t mean the cake will actually taste good.

Let me make clear to the programmers among you that I am one of you, and I know what it takes to implement a flowchart, and that the details aren’t so peachy—my Waring parameter estimator is giving me buggy output and I can’t work out why. But none of those hard details go into a patent application. Here is a surprisingly frank patent attorney explaining what is currently required: “Everything that goes into a claim can almost be correlated back to a simple flow chart, so you can get protection at a fairly high level. And, of course, that’s our goal, to get as much as we can.” (from page 137 of this FTC hearing transcript) So it comes back to the inventors trying to get as much as possible for as little as possible, which means an incredibly broad patent.

The courts and the USPTO have been delightfully polite in obliging them, because we need to encourage innovation. Here is Dan Burk of the U of MN law school doing his patent judge impression: “Tell us that it’s a compiler; tell us that it’s a spreadsheet. We’ll assume that you can write the code. You don’t need to tell us what the code is. You don’t need to give us a flowchart, don’t

need to give us any indication of how you do it, just tell us its function.” (p 134 of the same transcript) As a result, an inventor can file a claim with the USPTO which states, ‘I have asked my computer to index recipes for soylent blue,’ and now anyone who wants to ask their computer to index soylent blue recipes must first get permission from the patent holder.

If we had not given the inventor any protection at all, would he have asked his computer to do that which he needed it to do? Yes, he would have—he or his client needed it done, and to a good programmer, asking the computer to do something is cheap—no need to invest in expensive solar panels. Yet by giving him a patent on the process of asking a computer to do certain things, we have given him control over a potentially huge swatch of things a computer can do. For example, a company which has no products, named Acacia, has a patent on the concept of digital media, and has sued nine digital cable companies for infringing its patents. [Cherry, 2004]

Since asking a computer to do something is cheap, the first to do so should not have broad protection. But we are giving them the market equivalent of a harem for very basic work. In cost-benefit terms, we’re getting ripped off. For this and many other reasons, society is best off when innovators in computer code get narrower protection—a properly enforced copyright on their code, instead of a patent on a broad range of concepts.

## References

Steven M Cherry. The patent profiteers. *IEEE Spectrum*, June 2004.