including bug fixes from the community: vernacular design. What is described by Raymond is not so different from the way that human creative endeavors have come into existence for most of our collective past. In fact “the ‘open source’ way is closer to how human creativity has always worked” — it’s the proprietary model of production that is new.

In *How Buildings Learn*, Stewart Brand dissects vernacular building and describes a process that predates the Linux community while sounding uncannily similar. In Brand's view, the vernacular process is a systematic framework for evolving concepts. A culture steadily culminates frameworks over time. Vernacular designers reuse invented common forms and methods for common, everyday tasks. Vernacular in this context means common designs by common people. Common features survive the passage of time when they are collectively understood as “good.” Over the years then, a vernacular incorporates more and more “good” features while eradicating “bad” ones. Contemporary design typically looks for new or unique solution to a given problem — this is counter-productive to creating traditions and building tested solutions over time, and it cannot easily tke advantage of this methodology.

To illustrate this in *How Buildings Learn*, Brand references the Cape Cod house. His included diagram is named “The Evolution of the Siasconset Whale House.” This chart visualizes what Raymond is trying to explain to us in *The Cathedral and the Bazaar* (Braithwaite also wants contemporary designers to understand this), that simple things released into the wild of a user community will grow, evolve, and change — and will evolve improvements. The improvements merge back into the computer code or architectural language. Repeat. Torvalds is a genius to Raymond not because he uncovered something new, but in that he managed to take an existing form of cultural production and accelerate and decentralize it so as to maximize merging fixes and improvements and features!

If new software can be designed and constructed this way, and vernacular cultural objects have always existed this way, what's so hard in contemporary visual design?

Free as in Freedom

I consider that the Golden Rule requires that if I like a program I must share it with other people who like it.
— STALLMAN IN GNU MANIFESTO

Richard Stallman first published the GNU Manifesto in 1985. In it, he described "Free Software," which was in opposition to proprietary software. Stallman's motivations for this were both pragmatic and utopian. He hoped to maintain the kind of environment expected from academic computer programmers: a vernacular-like model where everyone builds upon everyone else's work. Stallman also was hoping to fight the tide of proprietary, softwares and tools he felt rising. Free software is predicated on the following freedoms:

```
A program is free software if the program's users
have the four essential freedoms:
   -    The freedom to run the program as you
wish, for any purpose (freedom 0).
   -    The freedom to study how the program
works, and change it so it does your computing as
you wish (freedom 1). Access to the source code
is a precondition for this.
   -    The freedom to redistribute copies so you
can help others (freedom 2).
   -    The freedom to distribute copies of your
modified versions to others (freedom 3). By doing
this you can give the whole community a chance to
benefit from your changes. Access to the source
code is a precondition for this.
       A program is free software if it gives
users adequately all of these freedoms.
Otherwise, it is nonfree. While we can
distinguish various nonfree distribution schemes
in terms of how far they fall short of being
free, we consider them all equally unethical.
```

Free software, as defined by Stallman, is ethical. It has a point of view. Its clear about where it stands. Braithwaite and Raymond do not deal with this in their discussion or analysis of "Open Design" and "Open Source" respectively. Students that were attempting to get a grip on their