Given this history, let us look at some of the ways in which a project like KDE tries to deal with issues that go beyond what a single developer, however talented, can solve. How can knowledge and experience be retained and used to maximum effect? Incorporating strategic thinking, how can a large, diverse group make difficult decisions and agree on steering toward the overall destination without jeopardizing the sense of fun? How can we incorporate the idea that decisions are made on equal footing among peers and the other aspects that have made KDE and other Free Software projects so successful? In other words, how to build a cathedral when everyone involved has a tendency to consider themselves, rightfully or not, an architect, and how to fill the role of architect without wearing a funny-looking hat. We start with looking at the contributors because we are convinced that Free Software communities are first and foremost social structures as opposed to technical ones. People are the most scarce and valuable resource we have.

There are different roles to fill for every project, and there is no authority that decides in the recruiting process. Every Free Software project needs a number of poster girls and boys and a large bunch of motivated, skilled, down-to-earth hackers, artists, administrators, writers, translators, and more. The one thing that seems to be a common denominator among all of these trades is that they all require a great deal of self-motivation, skill, and self-guidance. To be able to contribute seems to attract many extraordinary individuals of all ages, from high school students to retirees. What makes them join the project is the fascination of being part of a group that creates tomorrow's technologies, to meet other people interested in and driving these technologies, and often to do something with a reason, instead of writing throw-away college papers.

A well-functioning Free Software community is about the most competitive environment to be in. Most commercial engineering teams we have encountered are way more regulated, and policies protect the investment employees have made in their position. Not so for Free Software developers. The only criteria for a certain code contribution's inclusion in the software is its quality and if it is being maintained. For a short while, a mediocre piece of code made by a respected developer may stay in the source code, but in the long term, it will get replaced. The code is out in the open, constantly scrutinized. Even if a certain implementation is of good quality but not close enough to perfection in a few people's perception, a group of coders will take it on and improve it. Since creation is what motivates most, Free Software developers constantly need to be aware that their creation is made obsolete by the next one. This explains why code written in Free Software projects is often of higher quality than what can be found in commercial projects—there is no reason to hope the embarrassing bits will not be discovered.

Surprisingly, few coders want exterior motivation in the form of recognition from users or the media. Often, they turn away from some piece of work shortly before it is finally released. As with marathon runners, it seems their satisfaction is internal—the knowledge of having reached the finishing line. Sometimes, this is misunderstood as shying away from the public, but on the contrary, it only underlines the disinterest in praise. Because of this set of personal values, in software projects that are really free, it is almost impossible to assign priorities to