Reputation Layers for Open-Source Development

(Position Paper for the 1st Workshop on Open-Source Software Engineering)

Hassan Masum
Department of Computer Science, Carleton University; Ottawa, Canada.
hmasum@ccs.carleton.ca.....www.carleton.ca/~hmasum

Reputation as the Currency of Open-Source Software

Open Source software development is often done by distributed teams of enthusiastic volunteers. Why do they bother? It's certainly not for the money -- rather, goals include self-actualization, cooperating with a high-performance team, meeting a current need, and building a reputation within the developer and user communities. Even in commercial open-source, **non-monetary motivations** are important. How do these other motivations operate, and how can they be managed to allow for large-scale development? The challenge is twofold: to create systems that encourage productive co-operation among developers, and that give users some assurance as to quality and reliability of the software.

The **reputation incentive** is arguably the most "extensible" motivation, in the sense that it could at least in principle be extended to very large groups as a directing and group-binding force. An existence proof of this claim is the scientific research community: although scientists certainly do appreciate having money, they are also strongly motivated by peer recognition of their research, which has been semi-institutionalized through the peer-review and publication process.

Clearly, winning the approbation of one's peers is a very powerful human motivational force. Why is it so common in the research community, and relatively less common elsewhere? Perhaps the main reason is that scholarly publications have come to be universally recognized as a form of currency: **a strong publication record is a valuable asset** to any intellectual, tradeable for coveted teaching and research positions, consulting or speaking engagements, and so on. In order not to be subjected to "inflation", the peer-review process ensures a minimum quality level; the quality of the conference / journal and the number of citations also provide judging criteria.

How could this reputational system be extended to other domains, and in particular the open-source software engineering community? There are several **key requirements**:

- Quality. The assigned reputation should correlate highly with actual quality of work, or the entire system will quickly lose credibility.
- Accessibility. The reputations should be easily accessible to any interested parties.
- Nonforgeability. Reputations must be very difficult to forge.
- Universality. The system should be widely recognized as valid within its domain.
- Persistence. Users should have faith that the system will remain valid for a long time.

Note that these qualities are very similar to those for a monetary system; in fact, I would submit that a reputation system is analogous to a monetary system in many ways.

Merits of an Open Opinion Layer

An open-source, extensible, distributed, searchable, authenticated database of opinions would be very useful for commerce, product evaluation, recommender systems, research, and even practical politics. Let's consider applying this idea to open-source: clearly, just having easy access to opinions of users on software quality would be quite valuable. This would be similar to having **persistent "online references"** to your work, like references from employers.

However, simple opinions of users have several problems, such as users who are either misinformed or deliberately misinforming. (Excellent practical examples of this are the reputation system on eBay, and the book reviews on Amazon.) This is why **"certification authorities"** arise in so many fields: having standardized and trusted sources for valuation of quality is a very useful service. Some examples include credit agencies, bond-rating agencies, consumer product reviews, Underwriters' Laboratory, and educational institutions. What would such an agency look like for software, and who would pay for it and maintain it? To be useful, it should assign standardized quality levels, although this is difficult to do at the level of individual programmers.

Another practical problem is **"credit allocation"**. In AI methods, one often has the problem of distributing positive or negative performance of some algorithmic component among its subcomponents, but this requires figuring out the effects of each subcomponent on the overall utility, which can be far from obvious. Similarly, once distributed teams grow past a certain size, how can credit (or blame) be objectively assigned to team members? Companies have elaborate methods for deciding what the contributions of individual employees are to the overall business, but this area of employee compensation is a notoriously difficult management problem to solve.

This problem of fair credit allocation is one of the main stumbling blocks to scaling up open-source software development. It is easy enough to have a list of those who contribute bug fixes to each release, but how about more fuzzy tasks like system design, debugging, co-ordination, etc? Virtual software teams tend to choose tasks which are easily compartmentalized, so that they can get credit for achieving some specific goal. But how can reputation be fairly assigned in software projects that may last for several years and frequently change programmers?

Finally, consider the similarity with creating **reward schemes for intellectual property**. Producers of IP would often like to have some tangible reward for their effort, beyond the dubious flattery of being used by many people. Strategies proposed include micropayments, the "Street Performer Protocol", completion bonds, ancillary revenue streams, subscription models, and so on, but it is still quite unclear which if any of these will create adequate revenue for content producers. And similarly to open-source software engineering, there is an open-source content movement underway involving projects like creating an open-source encyclopedia (www.nupedia.com). The strategies used by these large collaborative projects to reward their members will likely be both instructive and useful for large software projects.

Further discussion and references on the open opinion layer idea are available in a paper I'm currently writing; a draft is available online (**www.carleton.ca/~hmasum/TOOL.html**).