A Robust Open Source Exchange for Open Source Software Development

Amit Basu

Cox School of Business, SMU
Dallas, TX 75275-0333, USA
abasu@smu.edu

Abstract. This paper addresses the development of mechanisms for the creation of OSSD exchanges that could be used by developers across any geographical range, as long as all the developers can interact via some open network infrastructure such as the Internet. The structure of these exchanges can range from public repositories such as Sourceforge.net to intra-organizational forums for software development within an enterprise. We examine in particular the structure of an exchange model based on protocols for a robust online marketplace.

1 Introduction

Open source software development (OSSD) thrives upon the ability to collaborate with other developers, and to reuse existing code developed by others. Thus, mechanisms for knowledge sharing and search are key resources for such development processes. Effective search requires mechanisms to learn about the availability of code segments that can be useful components in system development, and to obtain those segments when relevant. Effective knowledge sharing requires mechanisms that are sensitive to identities, roles and needs of each participant in the collaborative processes in OSSD.

This paper addresses the development of mechanisms for the creation of OSSD exchanges that could be used by developers across any geographical range, as long as all the developers can interact via some open network infrastructure such as the Internet. The structure of these exchanges can range from public repositories such as Sourceforge.net to intra-organizational forums for software development within an enterprise.

The different types of exchange or repository vary in terms of their support for key processes, and the paper surveys some of the key differences. It then examines one specific type of exchange in particular. The key feature of this exchange, which we call a robust open source exchange (ROSE), is that it enables individuals in specific roles (and groups) to interact in a way that provides them full control over disclosure of information, including identity information. At the same time, it provides robust mechanisms for accountability, so that anyone attempting fraud and/or deception can be reliably disclosed.

While the ability to withhold identity and information may seem counter to the open exchange philosophy underlying OSSD, it has some significant merits when

Please use the following format when citing this chapter:

Basu, A., 2006, in IFIP International Federation for Information Processing, Volume 203, Open Source Systems, eds. Damiani, E., Fitzgerald, B., Scacchi, W., Scotto, M., Succi, G., (Boston: Springer), pp. 99-108

implemented appropriately. The key consideration is that each participating individual controls their information and its disclosure, rather than the exchange itself, its owner, or any third party. Also, such an exchange provides a vital qualification procedure that can promote greater confidence in knowledge sharing and accountability.

2 Key ideas underlying OSSD

Perhaps the best characterization of open source software is in terms of the Open Source Definition [Perens, 1999], which lays out the following features that a program must have to qualify as an open source program:

- Free distribution
- Availability of source code
- Creation and distribution of derived works
- Integrity of each author's source code
- No discrimination against persons or groups
- No discrimination against fields of endeavor
- Distribution of license to all parties who obtain the program

On the other hand, in the enterprise context, while the merits of OSSD are desirable [Persson et al, 2005] the following considerations are important determinants of application development strategy:

- The ability to maintain control over intellectual property as well as applications that are strategic.
- The ability to ensure that developers are trustworthy and have no malicious intent.
- The ability to acquire necessary software development resources without disclosing identities and purposes to the general public (or to competitors).
- The ability to hold developers accountable for their work.
- The ability to set up development projects with schedule, quality and functionality stipulations that can be monitored and controlled.

These two sets of features are completely consistent with an approach that factors in reliability and security as requisites of the software development environment. In the setting of enterprise applications, and particularly with specialized applications having limited applicability across a broad population, an interesting approach to consider is an exchange, which allows providers and users of open-source software to find each other. Note that users may themselves be developers, and also note that the exchange may involve payment or not. Furthermore, the exchange may also be used to assemble the relevant distributed development team, in which specific individuals

or groups are assigned roles such as coder, maintainer, GUI Designer, documenter, etc.

Related Work

The idea of using market mechanisms as a basis for either the development or execution of computer programs is not new. In the realm of program control and execution, one of the most interesting approaches is based on the notion of agoric systems (from the Greek work "agora", which refers to a meeting place or market) [Miller and Drexler, 1988]. Computer-based systems and programs can be organized as agoric systems both in the small and in the large. The basis for the former is that software objects encapsulated with rational decision-making methods, can achieve meaningful execution of computer-based systems, when allowed to interact within the structure of a decentralized control mechanism that functions according to the rules of a well-structured marketplace. These rules respond to varying priorities of the autonomous objects, orderly contention for scarce resources such as processors, memory, storage and channels. This conceptualization is not directly related to software development. However, agoric systems can also be modeled in the large, namely at the level of the collection of resources that together construct a program or system. In other words, a collection of developers cooperating with each other in a democratic fashion can also be organized to interact according to the rules of a marketplace. In traditional software development, some of the insights derived from such approaches can be applied to the organization of third-party software development, or "contract programming" and outsourcing. Examples of economic models of such contexts include [Whang, 1992], [Whang, 1995], [Gopal, 2003].

There have also been a number of papers on market-based models for OSSD, on the notion that OSSD inherently relies upon a highly distributed and decentralized organizational model. One of the most interesting perspectives on this issue is presented in [Raymond, 2001], in which the hierarchical control structure of proprietary software development is contrasted with the more "bazaar"-like market-oriented model of OSSD.

Types of Exchanges for OSSD

Perhaps the earliest form of online exchange for OSSD was the online bulletin board and list-servers. These are largely un-moderated sites that allow relatively free access and participation, with relatively few controls and/or rules. While these are useful and inclusive, they don't scale well, and can easily be corrupted with irrelevant and/or unqualified contributions and even disruptive content. Also, the primary focus of these sites is on email-type interaction, and theus they do not have cataloged repositories for code, tolls, etc.

A popular approach to collaborative software development is the use of community-based software exchanges such as GNUenterprise.org [Scacchi, 2005]. In these community sites, there is little or no central control, although there are roles and protocols for interaction and participation. These have largely evolved from online bulletin boards and list-servers, and are largely directed at individual developers.

Another, related type of exchange is a community-oriented exchange that is developed by an OSS vendor. Strictly speaking, many of the exchanges that position themselves as community sites fall into this category. These include SourceForge.net, Eclipse and NetBeans.org [Jensen and Scacchi, 2005b]. The problem with such exchanges is that while the sponsorship of a major stakeholder (the vendor sponsor) promotes participation by individuals, the existence of competitive threats discourages institutional participation, by the sponsor's competitors for instance.

In all these approaches, the primary focus is on the individual developer/coder, rather than institutional participation. Therein lies a major challenge for moving OSSD from the fringes of mainstream software development to a forum and approach for enterprise-level and strategic software tools and applications. It is this challenge that we attempt to address with the proposal in this paper, namely the idea of a Robust Open Source Exchange.

3 A Robust Open Source Exchange (ROSE) Model

In this section, we describe a model for a robust open source exchange (ROSE) based on a set of protocols developed for robust online marketplaces, in [Kalvenes and Basu, 2005]. The protocols were designed to support the following features:

1. Participants in the marketplace have to qualify, through an authentication process conducted by a trustee. The qualification process can also include multiple levels (akin to credit ratings), so that traders can participate in a particular transaction (say at a given value ($) level) only if they are qualified to do so.
2. Buyers and sellers can transact through the marketplace without disclosing their identities or the details of their trades to anyone else, including the operator of the marketplace.
3. Since the marketplace operator has no competitive advantage over other traders, both the operator as well as its competitors can participate in transactions without fear of disclosure of transactions or strategies.
4. Although transactions can be anonymous to everyone other than its participants, any trader who commits fraud can be held accountable and identified by the marketplace operator and trustee.
5. Trader performance in the marketplace can be rewarded, so that trader qualifications can be modified over time.

The OSSD context is different from the transactional marketplace context described above. However, there are important similarities as well. Each of the above features can be reexamined in the OSSD context as follows:

1. Participants in a ROSE have to establish their credentials and be authenticated. This authentication process can also be used to qualify the participant for specific role. For instance, a relatively inexperienced programmer may be qualified to be a coder/contributor, but not a project lead or a tester/SQA (software quality analyst). Note that this in no way violates the spirit of the open software development "bazaar" [Raymond, 2001]. It merely ensures that users of code can trust that the providers are competent.
2. Users can post projects/requirements without disclosing who they are, and providers can bid for the contracts anonymously. At the same time, the qualification system ensures that only qualified bidders can post bids. The bids can be not only for code contribution, but also for other roles (e.g., project lead, tester, maintainer, etc). This is an important consideration if OSSD is to effectively penetrate the corporate software development market. While companies may be open to sharing code and related resources with the software development community, they are unlikely to want to share information about their software development needs and efforts with competitors. By keeping its identity secret during the negotiation phase of engaging external developers, a company can avoid prematurely revealing strategic intents to their competitors. On the other hand, this feature may also be very important for developers. For instance, many OSS developers are professionals who are employed by firms that are not supporters of OSSD, or are committed to proprietary platforms and systems. Such individuals may not be able to participate and contribute to OSSD if their companies knew of this. Protecting their identities while at the same time supporting authentication of their technical capabilities and credentials may be a necessary condition for their participation.
3. The privacy mechanisms and prevention of information asymmetry enables anyone to set up a ROSE, including an entity/firm that is itself a software developer. This is also important, since it has traditionally been difficult for enterprises to achieve the dual goals of having both broad participation in an exchange (which is easier when potential participants know that the exchange will give them access to large and important entities) and prevention of competitive exploitation (the threat of which is greater when a potential participant's competitor is the owner/operator of the exchange).
4. Users can be assured that any provider who bids on a job and wins a contract cannot refute on the commitment even if their identity is not disclosed to the ROSE authority at transaction time. The authority can identify the errant participant through a robust protocol. At the same time, the same protocol cannot be used improperly by the authority itself, without disclosure to the community. Once again, this is an important capability. If companies and individuals are to trust the exchange as a reliable means of connecting with qualified partners, there has to be adequate accountability. At the very least, anyone who violates the

codes of conduct must be identifiable, and held accountable. A common assumption in OSSD is that the openness of the community facilitates poor quality to be detected and problems to be resolved very efficiently due to the viral nature of the development process. While this may be acceptable for certain types of applications and projects, it is inadequate for enterprise-level projects. Therefore a positive feature of the ROSE approach is that it provides support for accountability and substantial recourse for dispute resolution.

5. As providers gain experience and credentials, they can be qualified at higher levels based on their track record at the ROSE. This is particularly important in a setting where individual participants want to offer their services through the exchange. The qualification level at which an individual enters the exchange may be different from their capabilities and credentials after gaining experience on one or more projects. The ability to re-qualify developers at higher levels thus is highly desirable. At the same time, if the identities of developers cannot always be revealed, the rewarding process has to work within those constraints. A positive feature of the ROSE approach is that this is supported as well.

We next describe how the following key processes can be implemented in a ROSE model:

- Registration and qualification of new participants
- Posting of a project RFP (request for proposal) by a prospective consumer
- Posting of bids by prospective providers in response to a project RFP
- Completion of a contract on a project
- Protection of intellectual property

In terms of the degree and scope of anonymity in the ROSE, any or all of the following modalities are possible:

1. Provider (P) and user (U) both disclose their identities throughout the process to everyone.
2. P discloses identity to everyone along with his offer, and U remains private.
3. Both P and U remain anonymous to the rest of the system throughout
4. Both P and U remain anonymous to both the system and to each other throughout.

While any and all of the above modalities may be desirable to support in a ROSE, in this paper we focus on the last case, where the highest level of privacy is desired. Each of the other cases can be supported by methods that are relaxations of the method proposed here.

Protocols for Anonymous Contracting

To start with, a new participant has to register at the ROSE, and be qualified at a certain level. This is done by the applicant obtaining a reliable digital certificate from a credible authority (e.g., a certificate authority firm such as Verisign or AT&T), and proof of qualifications in the form of a resume, transcripts, and endorsements from employers and/or customers. These are provided electronically to a trustee entity. The trustee is a firm or entity that works with the exchange operator, but may be totally independent. The trustee examines the application and supporting materials, and then provides the applicant with a software module (or client-side applet) called a certificate management system (CMS). This module resides and runs on the participant's computer, and enables them to participate in the exchange. Note that while we assume that the CMS is resident on a specific client machine, in principle it could be a mobile module that the participant could keep with them and attach to any machine that they use to access the exchange.

The CMS then generates a number of digital certificates for the participant for use in online transactions on the exchange. These certificates identify the participant, and include their credentials. The CMS blinds these certificates [Schneier, 1996], and then transmits them to the trustee, who then selects an arbitrary but substantial number of the certificates for examination. The trustee requests the blinding key for these certificates from the participant's CMS, and then un-blinds and examines the certificates (opening the certificates with the public key from the participant's original digital certificate). If satisfied that the certificates are all authentic and valid, the trustee then signs the remaining blinded certificates, and returns them to the participant. The participant's CMS then un-blinds the signed certificates, which can then be used as validated certificates for transacting on the exchange. Note that this process ensures that only authenticated entities can participate in the exchange, but neither the trustee, nor the exchange operator can track the participant's behavior on the exchange, since the trustee does not know which specific certificates are being used in each transaction (or who they belong to), and the exchange operator cannot open the transaction certificates, since it does not have the relevant public key.

During the transaction process, each participant (whether a developer bidding on a project or a company offering a project) provides a valid transaction certificate with each message sent to the exchange. These interactions between the different parties can be in the "public domain", in the sense that they are all posted on the exchange and are visible to all participants accessing the exchange and participating in that transaction's negotiation. When at some point there is a convergence between the relevant parties to a commitment to a contract, the parties can exchange signed messages that allow them to continue further communications in a fully attributed manner (i.e., they can identify themselves to each other).

A key feature of this process is that each of the parties has full control of how much information they divulge about themselves, and to whom, as well as when. This is an important consideration in an exchange where control of privacy may be a key constraint.

In the event of any conflict or misconduct, the other party can ask the exchange operator to identify and confront (and/or prosecute) the responsible party. Clearly, this is difficult to do in a setting where neither the trustee nor the operator knows the identities of the parties in any individual transaction. However, we have developed a robust method for this, in [Kalvenes and Basu, 2005]. It is robust, but expensive, since it involves a possibly large number of participants. In the commercial marketplace context, this can be addressed effectively by requiring each participant to put up a substantial escrow deposit held by the marketplace operator. However, this may be impractical in the OSSD setting. In this case, an alternative "penalty" may be blacklisting by the community and expulsion from the exchange.

Another consideration is the revision of developer qualifications based on performance. Again, using techniques developed in [Kalvenes and Basu, 2005], each developer can be given tokens by the exchange and/or the customer for each successful project, and can redeem these tokens with the trustee to revise their credentials.

Additional Considerations for OSSD

The above approach provides an innovative way to build and operate an online exchange for OSSD. However, there are some additional considerations in a ROSE setting that are worth examining. For instance, a big component of the value proposition of any software exchange is the repository of OSS code that is developed by participants, and which can be revised and further developed by other participants. As it turns out, support for this is completely consistent with the ROSE model. Note that the privacy concerns that motivate the ROSE model are driven by the strategic implications of tying applications to the companies that commission them, and the possible conflicts of interest that might constrain developers. The code itself can be easily maintained in an open repository, and issues of copyright can be supported by tagging the code with the certificate of the developer.

Another consideration in the OSSD environment is that projects involve multiple roles, and thus the relevant interactions are not always bilateral, between developer and customer, but possibly multilateral. Furthermore, this multilateral communication may have to be maintained throughout the development process. An interesting question is the extent to which such communication and collaboration would be possible without the different developers, testers, GUI designers, porters, etc. [Yeates, 2005] knowing each other.

Conclusion

In this paper, we propose a model for an online exchange that could be used to support OSSD within a large and distributed community of both developers and user entities. It attempts to address some key concerns about OSSD as it moves from the

fringes to the mainstream of software development at the enterprise level. It is intriguing to consider the use of such a model for an open source exchange. While many of the features of the ROSE model suggest a "closed" environment without the community benefits of more typical OSSD environments, it is actually possible for both a ROSE and a public repository (such as sourceforge.net) to coexist within the same context. In other words, the ROSE can be used to facilitate search, authentication, valuation and contracting, all of which are key to having a robust and reliable enterprise development environment. At the same time, once the development team is assembled through the ROSE, the development process itself can be facilitated by a public repository.

References

1. Anandasivam Gopal, Konduru Sivaramakrishnan, M. S. Krishnan, Tridas Mukhopadhyay, "Contracts in Offshore Software Development: An Empirical Analysis", *Management Science*, vol: 49, no. 12, 2003, 1671-1683.
2. Justin R. Erenkrantz and Richard N. Taylor, "Supporting Distributed and Decentralized Projects: Drawing Lessons from the Open Source Community", *Proc. 1st Workshop on Open Source in an Industrial Context*, Anaheim, California, October, 2003.
3. Chris Jensen and Walt Scacchi, "Collaboration, Leadership, Control, and Conflict Negotiation in the NetBeans.org Software Development Community", *Proc. 38th. Hawaii Intern, Conf. Systems Science,* Waikola Village, HI, 2005.
4. Chris Jensen and Walt Scacchi, "Experiences in Discovering, Modeling, and Reenacting Open Source Software Development Processes", *Proc. Software Process Workshop*, Beijing, China, May 2005.
5. Joakim Kalvenes and Amit Basu, "Design of Robust Business-to-Business Electronic Marketplaces with Guaranteed Privacy", working paper, Cox School of Business, SMU, 2005.
6. Martin Michlmayr, Francis Hunt and David Probert, "Quality Practices and Problems in Free Software Projects", *Proceedings of the First International Conference on Open Source Systems,* Genova, 2005.
7. Mark S. Miller and K. Eric Drexler, "Markets and Computation: Agoric Open Systems", in Bernardo Huberman (Ed), **The Ecology of Computation**, Elsevier Science, 1988.
8. Bruce Perens, **Open Sources: Voices from the Open Source Revolution**, O'Reilly Media, Inc., 1999.
9. Anna Persson, Brian Lings, Bjorn Lundell, Anders Attsson and Ulf Arlig, "Communication, coordination and control in distributed development: an OSS study", *Proceedings of the First International Conference on Open Source Systems*, Genova, 2005.
10. Eric S. Raymond, **The Cathedral & the Bazaar : Musings on Linux and Open Source by an Accidental Revolutionary**, O'Reilly Media, Inc. (2001).

11. Robert J. Sandusky, "Software Problem Management as Information Management in a F/OSS Development Community", *Proceedings of the First International Conference on Open Source Systems*, Genova, 2005.
12. Walt Scacchi, "Open EC/B: Electronic Commerce and Free/Open Source Software Development", *Proc. 5th Workshop on Open Source Software Engineering,* St. Louis, MO, May 2005.
13. B. Schneier, **Applied Cryptography**, John Wiley, NY, 1996.
14. Whang, S., "Contracting for Software Development", *Management Science*, vol. 38, no. 3, 1992, 307-325.
15. Whang, S., "Market Provision of Custom Software: Learning Effects and Low Balling", *Management Science*, vol. 41, no. 8, 1343-1357.
16. Stuart Yeates, "Roles in Open Source Software Development", OSS Watch, University of Oxford, 2005.