2008

Open Source Software in Complex Domains: Current Perceptions in the Embedded Systems Area

Bjorn Lundell
University of Skovde, bjorn.lundell@his.se

Brian Lings
University of Skovde, brian.lings@his.se

Anna Syberfeldt
University of Skovde, anna.syberfeldt@his.se

Open Source Software in Complex Domains: Current Perceptions in the Embedded Systems Area

Björn Lundell
University of Skövde, Sweden
bjorn.lundell@his.se

Brian Lings
University of Skövde, Sweden
brian.lings@his.se

Anna Syberfeldt
University of Skövde, Sweden
anna.syberfeldt@his.se

Abstract

With Nokia's 770 and N800 Internet Tablets heavily utilising Open Source software, it is timely to ask whether – and if so to what extent – Open Source has made ingress into complex application domains such as embedded systems. In this paper we report on a qualitative study of perceptions of Open Source software in the secondary software sector, and in particular companies deploying embedded software. Although the sector is historically associated in Open Source software studies with uptake of embedded Linux, we find broader acceptance. The level of reasoning about Open Source quality and trust issues found was commensurate with that expressed in the literature. The classical strengths of Open Source, namely mass inspection, ease of conducting trials, longevity and source code access for debugging, were at the forefront of thinking. However, there was an acknowledgement that more guidelines were needed for assessing and incorporating Open Source software in products.

Keywords

Open Source Software, embedded systems, qualitative study.

Introduction

The amount of software in products continues to grow apace. Much of this software is non-differentiating (Engelfriet 2007; COSI, 2005; West, 2007), so whilst expensive to develop and maintain it offers limited commercial advantage. Recently, there has been a resultant trend to look to Open Source Software to fill "the gap between innovation and product" (Engelfriet 2007). In a recent keynote address at Gartner Open Source Summit 2007, Mark Driver, Research Vice President of Gartner, is quoted as saying that by 2011 at least 80% of commercial software will contain significant amounts of Open Source code (Brodkin 2007). The commodification trend is noticeable not only for the amount of commodity software incorporated in products but increasingly for its creep up the software levels. West (2007) notes, for example, that Amazon's goal is to utilise commodity hardware, operating system, DBMS and application software, concentrating only on application services for differentiation.

The secondary software sector is that in which software is used as a component in other products, for example embedded software in the automotive industry. Hoepman and Jacobs (2007) suggest that different considerations may apply to this sector. One of these considerations is likely to be the need for long-term availability of products: for example, support of Airbus A300 will last for 78 years (Robert 2007). According to the FLOSS impact study (Ghosh 2006) the sector is large in Europe, and in it Open Source software "may have a particularly important role to play". This is a sentiment strongly supported by findings from the EU FP6 CALIBRE project, namely that, in the area of embedded systems, Open Source software is "fast becoming dominant" (Fitzgerald 2007). However, this interpretation is not straightforward. Most emphasis in such studies is placed on the sector's use of an embedded Linux kernel – that is, on Open Source software adoption at a low level in the software architecture. A more cautionary note is sounded in one of the CALIBRE practitioner interviews, in

which a lead engineer in a company heavily involved in the sector expressed a belief that, other than at the operating system level, Open Source software in the sector "is still very much at the trial stage".

There are clear exceptions. Nokia's 770 and N800 Internet Tablets heavily utilise Open Source software, which they consider "well suited for consumer devices" (Jaaksi 2007). A question arises, however. Is there any evidence that the general sector is changing in line with the overall trend, which is towards Open Source software adoption at all levels in the software architecture? We have explored this question through a qualitative study into perceptions of Open Source software amongst developers of embedded systems software as an example of a complex domain of application. We reason that this should highlight any early signs of movement towards broader adoption of Open Source software.

Research Approach

In this paper, we report on a qualitative study conducted with Swedish practitioners in companies developing software for the embedded systems market. Two companies were selected: one within the secondary software sector, being a manufacturer of high-end consumer products, and the other a software company contracting to the secondary software sector and specialised in the embedded systems domain. The study was with practitioners within the two companies known to be active in the development of embedded software. The business model of the software company means that it is common for consultants to be seconded to projects within client companies. Hence their experience spans many contexts within the embedded systems domain.

Data collection was based on the results of telephone interviews conducted in Swedish (the native language of all interviewees and the interviewer) and selectively transcribed. All quotations used in the analysis phase and reported in the paper are translations into English. A total of nine interviews were conducted, ranging in time from 45 to 90 minutes and resulting in 30 pages of transcribed interview data. Interviewees were selected on the basis of being actively involved in embedded software development. All but one holds a senior position.

The qualitative techniques used are designed to lead to richer information on the phenomenon studied, but do not allow any claim that the results are representative of organisations generally. However, the FLOSS project (FLOSS 2002) suggests that Swedish companies lag somewhat behind those in the UK and Germany in their uptake of Open Source software, so the levels of interest reported here may under-represent those held more generally within the EU.

The interviews were conducted over a two month period, and based on a number of open questions – a sub-set of which had one or more follow-up questions depending on the initial response. The questions explored two major dimensions of Open Source activity explored in the FLOSS project (experience of Open Source products, and interaction with OS communities); and a further dimension explored in the ITEA-project COSI (2005) (assessment of Open Source products), see, for example, Ayala et al. (2007).

Specifically, open questions for the analysis reported in this paper explored the following issues:

- What Open Source products are being used at the company and what has been the experience of using these products? Further, what has been the general perception of the quality of Open Source software? This explores the experience of Open Source software in the embedded systems industry.
- What methods for reviewing the quality of Open Source software are in use at the company, and do these recognise specific application areas in which Open Source software can replace proprietary software? This explores approaches for the assessment of Open Source software before adoption.
- Has there been interaction with any Open Source project, and in particular has any code modification been developed to increase the quality of Open Source software that has been adopted in the company? This explores interaction with the Open Source communities.

We presented our initial findings at the 1st International Workshop on Trust in Open Source Software (TOSS). In this paper we report on an extended data collection and a more comprehensive analysis of the results.

Current Perceptions in the Embedded Systems Industry

We analysed the transcripts of the interviews and grouped the responses around the areas of questioning described above. This gave us three broad categories of analysis, related to Experience, Assessment and Interaction, which are reported in the sections which follow.

Experience of Open Source Software in the Embedded Systems Industry

All of those interviewed were currently active in projects developing embedded systems software, and most had extensive prior experience of working in the area. We group the comments around four broad classes of Open Source product used.

Unsurprisingly, the most common experience of using Open Source software was with Operating Systems and in particular (but not only) Linux. It was felt to be simple to work with, stable, and to offer the required functionality; there was also significant trust expressed in the community behind it. Linux was currently being used both as a development platform and as a target platform. The common opinion was that it can replace proprietary operating systems. In several projects at the software company, versions of Linux have been preferred to proprietary operating system alternatives in the products developed. The experience is that Linux works very well, in one case replacing a proprietary option: "[we] bought an operating system that we intended to use that was not Open Source, but it soon showed that it lacked support for important aspects for us. We soon had to abandon that operating system. However, Linux had support for what we needed. The Open Source option was a better alternative than the commercial."

The second major area of Open Source use was compilers and debuggers, specifically C++ compilers. The GNU tool chain, and in particular GCC, was used extensively. The quality of these tools was considered to be excellent, superior to proprietary competitors. This was partly attributed to the tools' longevity, but also to their well-designed architecture. According to one consultant he had experienced no bugs in the GNU compiler but had encountered several problems with a proprietary alternative.

A number of libraries had been utilised, including Zlib for compression, various libraries for implementing network protocols, and Xerces for parsing XML. Only with Xerces was there considered a potential problem: it "consumed memory". The system was therefore constructed in such a way as to allow easy reconfiguration should resource usage become a problem. Interestingly, a commercial library had been tried for network protocols by one project leader, but support for it had been "terrible" and so an Open Source alternative had been sought and used.

Open Source development environments are currently less prominent in the embedded systems area. However, Eclipse is beginning to make an impression: "it has the functionality required, and it behaves well". There is an acknowledged learning threshold, but the benefits are considered worth the effort. For example, it "has good support for compiling, helping to localise errors in the code; this kind of support is lacking in the commercial tool we use today". One senior software engineer went so far as to say that the framework "is becoming the best alternative for the embedded systems area". Building plug-ins for Eclipse instead of building complete development environments from scratch is also seen as more cost effective.

There are, however, less satisfactory experiences also recorded. For example, CVS and subversion for version handling, even when backed up with other products such as Mantis, work only reasonably well: "they do what they were designed to do" – which by implication is not enough. They were compared unfavourably with such proprietary tools as Serena Dimension, which was considered to better integrate the required functionality. Tellingly, however, the project team selected CVS; the team was small, and CVS required a smaller learning curve than Serena. Now that the project is large the cost of moving from CVS would be prohibitive.

Open Source UML modelling tools are also not currently considered to have sufficient quality to fulfil the requirements of the software company for complete code generation using Model Driven Development. In some cases the desired features were not supported, and in other cases the general quality of the tools was poor.

One final example of Open Source software adoption is interesting for a different reason. The product manufacturer had commissioned software for html rendering which contained an Open Source component in the form of Mozilla SpiderMonkey. They were happy to accept the package and delegate to the provider the responsibility for implicitly guaranteeing the quality of the Open Source component.

The use of a third party supplier to mitigate exposure to Open Source was not in general considered a necessary option. The respondents directly downloaded software themselves and configured it for use. It was noted that configuration could be time-consuming as, unlike commercial software, Open Source software was often not "load and go" and did not come with simple configuration Wizards.

Assessment of Open Source Software in the Embedded Systems Industry

No specific assessment procedures were reported for reviewing Open Source software. The software is usually simply downloaded and tested. The code is not routinely reviewed and no formal tests are performed. The number of users of the software and the activity in the development project are considered. A number of interviewees expressed the need for guidelines on how to review Open Source software at different levels. It was felt that a structured way of verifying the quality

of Open Source software would help the developers to minimise risks. However, a number of strategies have evolved. We loosely categorise these to structure the arguments put forward.

Trialability is a term that has been coined to represent the ease with which developers can trial a piece of software. If a large licence fee is demanded, then this militates against trying it unless there is a compelling reason. Clearly the cost threshold for a trial of Open Source software is low: measured only in terms of the time to download and configure it rather than in budgetary costs. This is seen as important to experimentation: "we would not have looked at Eclipse if it had cost to experiment with it. We see that it can offer added value, but as we have bought other development environments we would not have bought another".

The most important criteria in the assessment strategies reported revolve around the Open Source community of the target software, including its user base. There was no perceived inherent quality difference between Open Source and proprietary software, so its user base (and hence its reputation) becomes the focus of attention. The extent to which the software is used plays an important role in whether it is considered of interest to the company: "If I see there is a large user base for certain Open Source software then I trust it more than a proprietary alternative, since then it is normally better tested and has a higher quality."

Software developed by too small projects or projects that are inactive is typically not introduced in the companies: "I feel safer in using software that is widely used." This is not simply true of Open Source software: "Fewer users of a proprietary system means it is likely there will be more bugs. Consequently it is likely that we ourselves will have to spend a lot of time on fixing the problem".

Checks are also made for references to other organisations that use the software, and for any review articles.

Product support is a major consideration when choosing to use software. When buying a proprietary product, there is a company behind the product that you can make support demands of. These may not be met immediately, "but at least you know that you will get it sooner or later." In fact a number of bad experiences of commercial support were reported, amongst them the "terrible" support for a library mentioned earlier. In general it is seen as very variable: "My experience of the support for commercial systems is that it is extremely dependent on which individuals are available in Sweden at the moment. It has even happened that there is no Swedish support at all… so I do not think there are any guarantees for commercial products either." This is backed up by a senior developer: "On one occasion the provider of [a proprietary Operating System] was in our office trying to find a bug, but they did not succeed."

Although expectations for support through Open Source communities were modest, experience was mostly positive. Direct interaction was not considered critical because "there is often an extensive amount of information published and there are others who have had the same problem and revealed their solution." Access to such information is often sufficient, so support issues are not seen as a reason for not using Open Source products. There is also recognition that third party suppliers can be used to mitigate support concerns: "Many consultants in the market can help us with issues on Open Source operating systems etc.".

There is an implicit quality threshold applied in any adoption situation. Primarily because of the support issue, "a slightly higher quality of Open Source products is required for us to use them, since there is no-one guaranteeing that the products actually work." What is meant by quality is not, however, clear and seems to relate as much to reputation as code quality. For example, on reviewing the code of some Open Source products one interviewee was not particularly impressed. However, experience showed that the software worked well and this was felt to be the important thing.

The issue of open versus closed code exercised many of the interviewees. "The problem with proprietary software is that you can run into bugs which cause a system crash and you cannot see what caused it, so cannot resolve what is wrong because you do not have access to the source code." This property was cited more than once as an advantage of Open Source software, hence increasing its attractiveness.

In a real sense, access to the source code is seen more as a guarantee: it is considered safer in that it is possible to modify the software independently of the software provider. However, in practice developers "want to avoid code modifications if possible" because of the initial and ongoing costs. It would probably require a critical application for the company to actually make such a modification, but the perception was that it is good to have the option. In the embedded systems domain such critical applications do exist: "We maintain some systems for a very long time (up to 15 years) and for such systems proprietary products are a problem if the software provider leaves the market. With Open Source software we get rid of this problem".

Interaction with Open Source Communities

In the software company, several interviewees reported on their interaction with Open Source projects, There have been different kinds of interaction, including: sending questions to Open Source projects, submitting bug fixes, and submitting code. Although response can be unpredictable, interviewees reported a good quality of response from questions posed to Open Source projects. For example, one interviewee commented on interaction with the community behind the GNU compilers. The interaction related to technical questions, and the response was considered fast and relevant. The questions and comments from the company were well received and quickly reacted upon (usually within one or a few days).

On the other hand, some interviewees commented on a lack of time for having extensive interaction with Open Source projects. One commented: "We cannot afford any slack time in our projects and therefore we mainly use software that we know works already from the beginning. It is too risky for us to wait for unpredictable feedback and corrections from Open Source projects." Further, when software is supported through a third party company, interaction would be with that company rather than directly with the Open Source community. There is another reason reported for not interacting: "The software has worked well from the start, so we have seen no need to."

Only one interviewee reported involvement as an Open Source developer outside company time, on a project for remote control.

Most of the interviewees had no experience of making modifications to Open Source software in order to increase its quality. One explanation offered is that when software is needed they only look for that which already works and fulfils the formulated requirements. However, one interviewee within the software company has contributed code enhancements to the community. They elaborated on their very considered interaction, commenting: "Often it is customisation or specialisation that we do. For example, when we adopt a boot loader that is Open Source and need to customise it for our very specific hardware, if the code that we have further developed adds anything to the software then we send it back to the Open Source project. But if it is very specific we do not, as no-one would have any use for it. Bugs that we find we report back; we do not solve them ourselves. On the other hand, we do workarounds if needed for our very specific hardware platforms."

In general, interviewees in both companies report that they prefer, if possible, to avoid making code modifications to the adopted Open Source code. For example, as one interviewee commented in the software company: "We do some customisation and add software on top of the operating system, but no big changes. We do not want to introduce any new errors, since it will be expensive and hard work to deal with. The volume of production of the [particular] product is too small – it does not justify developing our own code."

Analysis

The best Open Source software is considered by many to be of very high quality (see, for example, Broersma 2005; Lundell et al. 2006; Thomas and Hunt 2004) and certainly that was the general opinion of those interviewed. This is based on the principle that wide testing with scrutiny of the source code means that few bugs remain. This does not mean that all sections of the source code are written to a high standard (as noted by Michlmayr et al. 2005). When judging possible Open Source contenders, therefore, it is perhaps unsurprising that the size and reputation of the community of developers and users behind it mattered a lot, as did the vitality of the development community. This corresponds with the first point in Abedour's (2007) "important guidelines" for high-quality Open Source software. It is also consistent with the findings of Crowston and Howison (2006) on the health of Open Source communities, although other issues such as the shape of the community did not feature in the responses made.

Any company adopting an Open Source product is effectively placing trust in the ability of a virtual organisation to deliver a quality product. It is therefore not surprising that users of Open Source products generally look to reduce risk and thereby their necessary level of trust. This seems to be achieved primarily through peer review (Raymond 1999) – relying on the reputation of a piece of software amongst its user base. Effectively, this relies on massive testing to guarantee quality, reducing the perceived risk. In the companies this can be complemented by pre-filtering for projects that have recognised individuals or companies prominent in their communities.

It could be thought that buying proprietary software brings with it a guarantee of good quality support. This was clearly not the universal experience expressed. In one case support was so bad that the product was dropped in favour of an Open Source solution. Neither was it found particularly attractive to obtain Open Source software through a third party in order to mitigate risk. Although considered acceptable, so too did the option of buying in consultancy if and when a problem arose. In fact most of the Open Source software was directly downloaded and tested by those considering its use, so direct reliance on the Open Source project community resulted. Reported experience was positive, both with the passive support offered by looking at others' bug reports and past topics in discussion forums, and with active support through responses to posted bug reports.

Trialability was an important property of Open Source software, because it allowed download and testing with no overhead and no budgetary implications. It was also possible to trial software outside the office; many developers had tried Open Source software first at home. This phenomenon is noted by Dedrick and West (2003) in their consideration of why firms adopt Open Source platforms.

Interestingly, no mention of Open Source licensing was made in the interviews, in spite of the recent high profile of the topic (see, for example, Linux Foundation 2007). It had been expected that, particularly for production platforms, licensing would have been a significant issue in selecting software for adoption, and would have been raised as an issue with respect to code changes. However, with most of the software mentioned coming under LGPL or Mozilla licensing, it is possible that no such complexities had yet been encountered. It was clear, however, that at least with one of the interviewees there was not a clear awareness of the different licences covering Open Source though to shareware.

The companies analysed cannot, in this respect, be considered mature adopters of Open Source software in that no explicit company business strategies were evident for Open Source, and there were no mature procedures for assessment and adoption of Open Source software in place such as, for example, the Business Readiness Rating of Wasserman et al. (2006). This was well recognised by the interviewees, more than one of whom noted a lack of defined process.

A recurrent theme has been the longevity of embedded software. In the case raised during the interviews, the software had to be maintained for 15 years; in the case of Airbus A300 it is a staggering 78 years (Robert 2007). It is well recognised that software vendors are not guaranteed to have such longevity, and certainly not any particular product. The attractiveness of Open Source software then becomes apparent: it is like having a software asset held in Escrow. Related to this is the expressed need to be able to access the code for debugging purposes when integrating software, an option not possible – or at least unaffordable for SMEs – with proprietary code.

Conclusions

In this paper we have reported on the views of selected practitioners in two medium-sized companies undertaking software development in the embedded systems domain, one a software consultancy company and the other a manufacturer of high-end consumer products. The practitioners selected were familiar with Open Source concepts, and had at least some experience of using Open Source software within embedded systems projects.

The level of reasoning about Open Source quality and trust issues was commensurate with that expressed in the literature. The classical strengths of Open Source, namely mass inspection and testing, were at the forefront of thinking. Interestingly, mediation through a third-party company was not considered imperative to reduce the risk of adoption; instead there was positive experience when relying solely on available information kept by, and direct interaction with the Open Source communities. However, the availability of consultancy around an Open Source project was seen as an attractive alternative.

However, the companies could not be considered as mature Open Source adopters. Firstly, there was no strategic, company-level policy on the role Open Source software will play in their core business, and no defined best practice for Open Source adoption. Secondly, there was limited interaction with the Open Source communities other than for feedback.

In summary, there was a high level of trust in Open Source products with large, well established communities of users, meaning that adoption in products was not seen as a problem; an appreciation of the relative advantages offered by Open Source software, and in particular longevity and source code access for debugging; but an acknowledgement that more guidelines were needed for assessing and incorporating Open Source software in products.

As a final observation, the study adds further evidence that the Eclipse framework is now "becoming a very widely used platform for embedded systems development tools" (Day 2006).

Acknowledgements

This research has been financially supported by the ITEA project COSI (Co-development using inner & Open source in Software Intensive products) (http://itea-cosi.org) through Vinnova (http://www.vinnova.se/). We gratefully acknowledge the two companies for enabling this research.

References

1. Aberdour, M. "Achieving quality in open source software," IEEE Software (24:1), 2007, pp. 58-64.

2. Ayala, C., Conradi, R., Sørensen, C.-F., Franch, X. and Li, Y. "Open Source Collaboration for Fostering Off-The-Shelf Components Selection," In IFIP International Federation for Information Processing, Volume 234, Open Source Development, Adoption and Innovation (Feller, J., Fitzgerald, B., Scacchi, W. and Sillitti, A., Eds.), 2007, pp. 17-30, Springer, Boston.
3. Broersma, M. "IDC: Quality drives European open source adoption," ITworldcanada.com, April 22 2005, <http://www.itworldcanada.com/Pages/Docbase/ViewArticle.aspx?id=idgml-8f87ddb3-bfe0-4b69&s=90323>.
4. Brodkin, J. "Open source impossible to avoid, Gartner says," Network World, September 20 2007, <http://www.networkworld.com/news/2007/092007-open-source-unavoidable.html>.
5. COSI. "Heterogeneous distributed development: Co-development using inner and open source in software intensive products," ITEA: Information Technology for European Advancement, October 2005, <http://www.itea-office.org/public/project_leaflets/COSI_profile_oct-05.pdf>.
6. Crowston, K. and Howison, J. "Assessing the Health of Open Source Communities," IEEE Computer (29:5), 2006, pp. 89-91.
7. Day, D. "Eclipse: Under The hood – Overview of the Eclipse open-source IDE framework," Embedded.com, April 3 2006, <http://www.embedded.com/columns/technicalinsights/184417260?printable=true>.
8. Dedrick, J. and West, J. "Why firms adopt Open Source Platforms: A Grounded Theory of Innovation and Standards Adoption, Standard Making: A Critical Research Frontier for Information Systems," MISQ Special Issue Workshop, 2003, pp. 236-257, <http://www.si.umich.edu/misq-stds/proceedings/145_236-257.pdf>.
9. Engelfriet, A. "Open Source and Open Innovation, Koninklijke Philips Electronics NV," handout: LinuxWorld Open Summit 2007, Stockholm, September 5 2007, <http://www.idc.com/nordic/downloads/events/linuxworld07/9%20-Arnoud%20Engelfriet.pdf>.
10. Fitzgerald, B. "The transformation of open source software," MIS Quarterly (30:3), 2007, pp. 587-598.
11. FLOSS. "FLOSS Final Report – Part 1: Free/Libre Open Source Software: Survey and Study," University of Maastricht, The Netherlands, 2002, <http://www.infonomics.nl/FLOSS/report>.
12. Ghosh, R.A. "Study on the: Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU," Final report, MERIT, November 20 2006, <http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf>.
13. Hoepman, J.-H. and Jacobs, B. "Increased security through open source," Communications of the ACM (50:1), 2007, pp. 79-83.
14. Jaaksi, A. "Experiences on Product Development with Open Source Software," In IFIP International Federation for Information Processing, Volume 234, Open Source Development, Adoption and Innovation (Feller, J., Fitzgerald, B., Scacchi, W. and Sillitti, A., Eds.), 2007, pp. 85-96, Springer, Boston.
15. Linux Foundation. "Leading Open Source Legal Experts Join Legal Foundation," The Linux Foundation, August 8 2007, <http://linux-foundation.org/weblogs/press/2007/08/08/leading-open-source-legal-experts-join-linux-foundation/>.
16. Lundell, B., Lings, B. and Lindqvist, E. "Perceptions and Uptake of Open Source in Swedish Organisations," In Open Source Systems: IFIP Working Group 2.13 Foundation Conference on Open Source Software – The Second International Conference on Open Source Systems (Damiani, E., Fitzgerald, B., Scacchi, W. and Scotto, M., Eds.), 2006, pp. 151-163, Springer.
17. Michlmayr, M., Hunt, F. and Probert, D. "Quality Practices and Problems in Free Software Projects," In Proceedings of the First International Conference on Open Source Systems (Scotto, M. and Succi, G., Eds.), 2005, pp. 24-28, Genova.
18. Raymond, E.S. "The Cathedral & the Bazaar," O'Reilly. hardcover ISBN 1-56592-724-9, October 1999.
19. Robert, S. "New trends and needs for Avionics Systems," ARTEMIS Conference, Berlin, May 2007, <http://www.artemis-office.org/DotNetNuke/Portals/0/Conference%202007/SYLVIE_ROBERT_AC_2007.pdf>
20. Thomas, D. and Hunt, A. "Open Source ecosystems," IEEE Software (21:4), 2004, pp. 89-91.
21. Wasserman, A.I., Pal. M. and Chan, C. "The Business Readiness Rating: a Framework for Evaluating Open Source," In Workshop on Evaluation Frameworks for Open Source Software (EFOSS), Como, June 10 2006, <http://www.openbrr.org/como-workshop/papers/WassermanPalChan_EFOSS06.pdf>.
22. West, J. "Value Capture and Value Networks in Open Source Vendor Strategies," In Proceedings of the 40th Hawaii International Conference on System Sciences – 2007, IEEE Computer Society, Los Alamitos, 2007. 10p.