An Initial Analysis on How Software Transparency and Trust Influence each other

Luiz Marcio Cysneiros[1], Vera Maria Benjamim Werneck[2]

[1]*York University – School of Information Technology, Toronto*
cysneiro@yorku.ca
2Universidade do Estado do Rio de Janeiro – UERJ-IME
vera@ime.uerj.br

Abstract

If we make a simple search on the internet for the definition of transparency there will be several different ones from optics to protocols. But most of the definitions will overlap among the notion that transparency is about how something is open enough to allow things to be deeply observed from different perspectives. Orthogonally, Transparency has been demanded in several different areas in our society. Governments are demanded to be more transparent, banks are being blamed for not being transparent and so on. In a world where software is already pervasive and where the internet is connecting individuals all over the world, software transparency seems to be not only a remote possibility but something we will have to deliver sooner than many have thought. This paper aims at showing that trust is one of the important features to achieve transparency although this trust can be sometimes misleading.

1. Introduction

Defining software transparency is itself a challenge. If we use a search engine looking for existing definitions we may find only a few results. Transparency software [14] tackles it from the business-level solution perspectives where it calls for business people to be able to monitor, manage, track and audit information subject to requirements. This would require among other things a very detailed monitoring on database transactions. In Eclipse webinars page [05] we can find a claim supporting Palamida for promoting software transparency that allows to answer questions as "what is in your code?". In the Economist [06] magazine it is stated that open software contributes towards transparency due to its openness and collaboration.

However, transparency can also be seen from non-technical perspectives as for example what a society perceives today as the need for transparency. For example on how their governments make decisions, how they spent federal money, how they judge and allocate priorities. Recent economic difficulties have raised critical issues such as how banks are not transparent in their processes and balances. It has also showed to which extent that this may hurt people's trust in them as well as how governments may allocate money. Nowadays software is pervasive and clearly is key for the daily operation of governments, companies, banks, etc. Putting that together with the cry for transparency in these organizations will likely demand software to be built to deliver transparency not only about the data it is manipulating but also on the software itself and what is the reasoning behind the data created and manipulated.

Leite states in [10] that for software to be transparent it has to allow the information it deals with to be transparent and it has to inform about itself, how it works, what it does and why. Key for accomplishing this scenario is that requirements be as readable for general stakeholders as well as for developer's stakeholders. We subscribe to this idea.

Another lesson we can obtain from the recent financial crisis is that trust is important for people to believe that the transparency they demand is indeed being delivered. Take for example the AIG affair. At first the majority of tax payers, although reluctantly, were trusting the money that government was

allocating to save AIG was being well used just to later learn that dozens of AIG top executives were getting 165 Million dollars in bonuses despite having led the company to bankruptcy. Another example can be drawn from a recent episode in the United States and in Canada that illustrate how trust without transparency can be misleading. Ticketmaster is facing a class action suite for redirecting buyers to a second site where tickets were up to 60% more expensive than they should. But since Ticketmaster was redirecting without making it explicit people would trust in the prices offered since they implicitly trusted Ticktmaster. If the software was transparent such situation would not have happened.

We think that trust is one important component to achieve transparency. By the same token transparency in many cases may be essential to assure trust. Sink exemplifies some of the reasons for that in [13]. He states that similar to how restaurant chains benefit from transparency by showing the customer's food being prepared in open view behind a window, trust and transparency in software is what removes the veil. He also points out that when people buy software there is an implied trust involved in this transaction.

From a corporate perspective, transparency is then important; if you can improve the relationship with your customers, they will provide you with feedback that will help to make a better product. However, the level of transparency has to be managed to ensure trust and also to remain competitive especially for the old-fashioned way of doing business that believes that their information and process have to be closed.

As Leite [10] and Yu [18], we consider trust and transparency to be Non-Functional Requirements, and as such they are likely to impact each other. Although at first sight we tend to assume that trust may only bring positive contributions to transparency and vice-versa we will show throughout the paper that this assumption may not hold sometimes.

This paper will analyse some of the interdependencies between trust and transparency showing how one can improve the other. We also show that sometimes implicit trust can be deceiving and misleading if transparency is not met. We do not intend this paper to explore all relationships between trust and transparency. In fact, we hope this paper will stimulate other researchers to deeply investigate it.

This paper is structures as follows: Sections 2 to 5 present different perspectives for looking at transparency and some of the requirements arisen from each of these perspectives if we are to build software with transparency as goal. Section 6 briefly discuss some of the things we learned during this work while section 7 concludes the paper

2. Open Software

We believe that the use of Open Software is linked to achieving transparency in software. One of the main reasons is because knowing that the source is available to be read by anyone would lead people to trust the software since it would be unlikely to find malicious code or malicious use of information.

According to Blankenhorn [04] the modularity of open source is its beauty, since traditionally the user community develops and recommends as a whole the components to install, or remove those which are unnecessary. This works according to Blankenhorn, because it is analogous to many social structures that have been similarly successful. He states:

> "The best economic systems are transparent. This doesn't mean they're not regulated. But you can see the regulations, see how they're enforced, and see the results. Everyone comes to the market with their hands open. Things are as fair as possible.
>
> The best political systems are transparent. Again this is not the same thing as free. But if you can change the law, and if the system for changing things works, with the same rules for everyone, and the popular will respected, then you have a good system.
>
> The same point is clear in software as in business and in politics. Transparency wins. This is the great lesson of the 20th century. All the more opaque systems -- fascism, communism, dictatorships military and religious -- failed the great test against more open, transparent systems of capitalism and democracy. The triumph of open source, then, is simply the lessons of history applied to software."

According to Baird [03] corporate enterprise is not ignoring the benefits of transparency. Modern systems evolve to support the user and include the best of both worlds: the interoperation of market-ready proprietary software in combination with custom open-source software (OSS) projects from a well-established community of dedicated developers.

Baird also states that part of the open-source community has been adopting development and business strategies from the proprietary world. Development is no longer haphazard, but in many cases driven by a roadmap with milestones and schedules. Naturally, that has led to a more organized development and eventually to a greater predictability for customers. Developers are more likely being paid to work on open-source projects. Baird points out that the processes undertaken to develop open-source and proprietary software are looking more and more alike.

Likewise, Baird writes that proprietary vendors like Microsoft, IBM, Sun and Oracle are increasingly employing open source measures to diversify their product lines. IBM has purchased OSS startups in response to lost market share, Oracle has offered free express versions of database clients, [07] and most have dabbled employing testers from the open-source development community early on in the development process to ensure interoperability of their products with OSS. [03].

The adoption (even if only in part) of open-source code and ideas tend to improve the way people look at those companies and increase how they trust in these companies. In such scenario perceiving transparency through trust may become a largely adopted practice in the software development community.

3. Software Purity

An interesting idea is brought by Meunier [11]. He defines a second concept, called software purity, which is a standard that software should live up to. He considers it separate from transparency, because a transparent product may disclose functions to users, but in being pure, among others properties, the software upholds a standard that establishes user trust and loyalty that data will not be mined and software will not be surreptitiously installed in the background. Garfinkel [08] conceives the idea of the "Pure Software Act," which, like the U.S. Pure Food and Drug Act of 1906, forces disclosure of the ingredients of software, and what side effects exist from using it, in an effort towards realizing consumer protection. Government-mandated icons would be included from wherever the software was obtained, and would indicate functions such as whether the program installs and runs during booting of a computer, if it modifies your operating system, monitors use of your computer, if it is stuck to registry files and unable to uninstall, etc.

4. Cryptography

In providing security, cryptography is a solution that some companies are adopting in some transactional applications like e-crash, e-vote. However, cryptography brings some transparency problems, because they can improve security especially in a data network transmission of e-commerce transactions by bringing lack of transparency [16].

Mercuri [16] discusses the cryptography approach used by Visa and Master Credit Card Systems that also distribute the service along different sites to implement security. This solution can improve security but it increases complexity and does not really bring human trustability to the system. The lack of transparency and the belief that cryptography does not always secure the data are some issues that start to appear in discussions for guaranteeing trust in some digital transactions.

Transparency can be viewed as inversely proportional to trust. This approach relies on the belief that security may remain in obscurity that only few people have the knowledge about. For example the cryptography approach solves the problem of data vulnerability during transmission along the network because only few people know about the cryptography algorithm.

So the management of risk seems to be the point to be analysed for improving safe and transparency systems on web services to deal with those conflicting goals and solutions. The optimal balance of those criteria is still difficult to arise.

5. Licenses and Data collection

Although we assume we reasonably understand what is behind licenses and data collection, it is interesting to note that sometimes we are led to a false trust and therefore to a false sense of transparency only because we assume things.

An interesting discussion about licenses can be found in [13] Among Sink's recommendations, we find : "Don't Annoy Honest People.". License

enforcement code, such as product activation codes, are necessary to ensure that the software was obtained legally. Sink [13] believes from personal experience that requiring licensing is often detrimental because software vendors do two things: “1. We fight a battle we cannot win. Those who want to cheat will succeed; 2. We hurt the honest users of our product by making it more difficult to use.” Although product activation mechanisms and license agreements serve as a good indication that you trust customers to use your product legally, they can often perform the opposite; if the validation code is buggy, you run the risk of denying honest, paying customers the ability to use the product, when there is more code that has to be maintained. This puts a strain on the customer relationship.

Sinc [13] states that trust and transparency should be demonstrated beyond simple terms-of-use license agreements. Though license agreements have legal standing, they state what a user can do, any purposes of information gathering, what the user is and isn’t allowed to do with the program (i.e. distribute it illegally), etc.

This is at least what a user assumes it to state. Instead, many license agreements may contain statements embedded in the middle of the text that would allow data collection that a client may not be comfortable with. But since we frequently trust most of these companies we assume the agreements will only contain what is expected from them. Adding to this situation the fact that reading the agreements can be time consuming and boring, we simply say yes without reading and therefore we do not realize we may be authorizing things we may not have authorized if we have read it carefully.

Spyware and adware actually thrive on this assumption. According to researchers at Berkeley University [09], it was found that people unwittingly install malicious software, believing that their operating system or antivirus protection will protect them. Then, after installing, they find the software to be malicious; they express regret at not reading the license agreements. This often happens because users will make a tradeoff between their level of desired privacy and security to achieve small monetary gains (i.e. free programs). Garfinkel’s article on transparency and purity [08] calls this the “art of deception” of user agreements. Users will give consent to data collection and manipulation by the text concealed in spyware and adware installation notes:

> “The text more-or-less spells out all of the covert tricks that these hostile programs might play on your system. Of course, hardly anybody reads these agreements. Nevertheless, the agreements effectively shield purveyors of spyware and adware from liability. After all, you can't claim that the spyware was monitoring your actions without your permission if you gave the program permission by clicking on that "I agree" button” [08].

Garfinkel [08] indicates that companies like Google are less deceptive where being less deceptive should mean slightly more transparent. Google’s PageRank feature integrated into Microsoft’s Internet Explorer browser collected user browsing history, but “Google goes out of its way to disclose this feature-when you install the program, Google makes you decide whether you want to have your data sent back or not. “Please read this carefully,” says the Toolbar's license agreement, “it's not the usual yada yada.”. Berkeley researchers designed experiments to test certain pre- and post- installation warnings that provide ample warning to the user about the information that would be used, in plain English, and found positive results. Having operating systems and legitimate software vendors increasingly use such measures for installation can be used to an advantage, being more open with their users should allow them to allay fears about what information is collected and gaining reputation as a reputable, transparent company.

Terms and conditions in installation notes often only specify in legal terms what rights the consumer is signing over to the author instead of describing in plain English what personal rights are being subjected to potentially controversial actions. As an example of a more common software product that has raised some privacy and trust issues, we use Apple’s iTunes. Following the release of iTunes 8.0, the Genius feature was added. It builds a list of recommendations in a user’s iTunes music library for songs that go together, and can make recommendations for new music [01]. This has revealed some mixed reviews about the product, specifically as violations of privacy due to collection of personal data combined with knowledge about musical tastes, and many are equating the tool to spam [02]. Genius suggestions can be seen as an example of gold-standard data mining, opines Joe Wilcox on his Apple Watch blog: “The real genius isn't the new feature, but how it generates lots of marketable, trendable data for Apple.” [15]. Moreover, there is the stipulation that in order to use Genius, user account information and possibly credit card

information is tied in to the data that is mined. On his cnetnews.com Digital Noise music blog, Matt Rosoff pushes aside this concern that a colleague has about this dangerous provision of information:

> "I know why Apple requires an iTunes account for Genius: if Genius recommends a song from the iTunes Store, Apple doesn't want to interrupt your potential impulse buy by making you enter a credit card number at that time. But the iTunes requirement and sign-up screens made my colleague feel like Apple was asking too much for what he was getting in return. Fair enough, I usually enter fake names and e-mails for newspaper sites that require free registration, and I don't give any information at all to certain Web sites. We all have different comfort levels." [12].

This might lead one to think: why should we have to falsify our information in the first place? How does one have a certain 'comfort level' with lying? Should Apple be required to clearly state the intentions of using Genius, as a means to market to the user and sell a product? One could answer that Apple is a business, and iTunes is their success of innovatively marketing music that it sells. They are not the only company making software freely available to download off their website. However, here Apple may be seen as exploring the fact that most users would trust it to promote a market ploy that was deceptive with its clients.

6. Discussion

We showed four different perspectives to transparency and trust in the previous sections. From there we can see that transparency can be achieved in many different forms. We can also see that trust and transparency may not always support each other. This was in fact an unexpected outcome for this work. At the beginning our assumption was that trust would always help transparency and vice-versa.

It is important to make a distinction between Full Disclosure and Transparency. Even if software indeed offers full disclosure, it may not be transparent at all. For example, in 2004 TiVo offered full disclosure when you used your Digital Video Recording equipment. It actually alerted consumers that some data collection may be made while you were using the equipment and that you could opt out from the data collection by making a phone call. Users were surprised later when they knew that TiVo was able to tell how many people paused and rewound Janet Jackson's exposing herself.

The issues mentioned above alone raise an interesting line of research. How transparency and trust may work contrary to user's interest? By the same token what are the situations where privacy may contribute to achieve trust and vice-versa?

As Leite states in [10] the need for transparency brings together the need for linking requirements to models. Therefore, the requirements community should pursue a framework that allows both forward and backward traceability having transparency as the key motivator for this traceability. We believe that the i* framework is one of the best candidates to support such framework. Leite has proposed to use it to tackle transparency while Yu [18] has already indicated that i* can model trust without creating any new concept carrying special semantics to represent trust.

Of course we only scratch the subject in this work. Clearly transparency may be achieved in many more different ways, many times using a combination of approaches and perspectives.

7. Conclusion

Since today more than ever transparency in many different levels of our society is becoming almost mandatory and software is ubiquitous, it seems likely that we will see an increasing need to achieve and show that our software is transparent. By the same token trust regarding both the software and the provider will be increasingly demanded by costumers.

This paper aims at introducing interdependencies between trust and transparency on software. It shows some issues where trust can either improve or hurt real transparency and vice-versa.

We do not intend to be exhaustive. Rather we intend to show that these two subjects and how they relate to each other are far from being understood. We hope that this work may trigger the interest of researches to pursue a deeper understanding on how trust and transparencies can be used to benefit the society and therefore meet its increasing demand for it. By the same token we expect to see an increasing number of work on how do we achieve transparency and how should we do it.

We are planning to express those ideas using the i* framework as well as to deeply investigate and model some situations that happens nowadays like the trust analysis found in Yorkoff, Yu and Lin [17].

8. References

[01] What's New in iTunes 8. Apple Inc. Retrieved 7 March 2009 available at <http://www.apple.com/itunes/whatsnew/>.

[02] Did you read the Genius Privacy Policy? It's a license to spam. Message posted to <http://discussions.apple.com/thread.jspa?threadID=1701192&tstart=0>.

[03] Baird, Stacy. "The Heterogeneous World of Proprietary and Open Source Software." ACM International Conference Proceeding Series; Vol. 351 Proceedings of the 2nd International Conference on Theory and Practice of Electronic Governance Cairo, Egypt 2008, pp:232-238

[04] Blankenhorn, Dana. (19 April 2005). Open Source Transparency. April 2005 <http://mooreslore.corante.com/archives/2005/04/19/open_source_transparency.php>.

[05]http://www.eclipse.org/community/rcpwebinars2006.php

[06] http://www.economist.com/business/displayStory.cfm?story_id=2054746

[07] Rand, Matt. 1 Open Source Invades the Enterprise. Forbes.com. November 2005 <http://www.forbes.com/2005/11/01/bow051101011.html>.

[08] Garfinkel, Simson The Pure Software Act of 2006. Technology Published by MIT Review April 2004 <http://www.technologyreview.com/computing/13556/page1/>.

[09] Good, Nathaniel S. et al. 3 May 2007. Noticing Notice: A Large-Scale Experiment on the Timing of Software License Agreements. Proceedings of the SIGCHI conference on Human factors in computing systems p607-616, 2007

[10] Leite, J.C.S.P and Capelli, C. "Exploring i* Characteristics that Support Software Transparency" Julio Cesar Sampaio do Prado Leite, Claudia Cappelli, in Proc. of the 3rd International i* Workshop pp. 51-54

[11] Meunier, P. "Software Transparency and Purity" Communications of the ACM, Vol. 51 Issue 2, p104-104, Feb2008.

[12] Rosoff, Matt. iTunes Genius and Privacy. October 2008 <http://news.cnet.com/8301-13526_3-10062590-27.html?tag=mncol>.

[13] Sink, Eric. Tenets of Transparency. February 2005 <http://www.ericsink.com/bos/Transparency.html>.

[14] http://www.transparencysoftware.com/

[15] Wilcox, Joe. iTunes Genius: Not So Smart. September 2008 <http://blogs.eweek.com/applewatch/content/itunes/itunes_genius_not_so_smart.html>.

[16] Mercuri, Rebecca, "Trusting in transparency", Communications of the ACM, Vol. 48 Issue 5, 2005, p15-19.

[17]Horkoff,J., Yu, E. and Liu ,L; "Analyzing Trust in Technology Strategies" , Proc. Int. Conf. on Privacy, Security, and Trust (PST'06), Toronto, Canada, 2006, pp

[18] E. Yu, L. Liu "Modelling Trust in the *i** Strategic Actors Framework'" Proceedings of the 3rd Workshop on Deception, Fraud and Trust in Agent Societies. Barcelona, Catalonia, Spain (at Agents2000), June 3-4, 2000.

ACKNOWLEDGEMENTS

This work is partially supported by NSERC grant number: 262148-05