Information technology suffers from a distinct lack of care with respect to adequate informed consent procedures. Computer users are commonly asked to consent to various things that could threaten their personal identity, privacy, and property, yet little care is taken in assessing whether the consent is truly informed. Some software even takes advantage of the confusion rife in informed consent procedures in order to install otherwise unwanted software on users’ computers (such as adware or spyware). End user licence agreements (EULAs) are a common example of these poor informed consent procedures, which have their basis in the inappropriate use of medical informed consent procedures, instead of being based on concepts that have been developed specifically for IT. In this paper, I outline some of the problems that need to be overcome in informed consent in IT, and present a more appropriate theory for informed consent (based on the Manson and O’Neill waiver of normative expectations theory). I will then look at some current and recent suggestions for improvements or practical implementations of mechanisms to deal with informed consent. Then, in the third section, I will firstly identify the normative expectations, establish a communication framework, and then suggest an agreement mechanism that greatly improves the potential for informed consent decisions for EULAs. Examples will be used to illustrate the process, but will not cover the details of the actual legal agreement. It will, instead, focus on delivering the content of this agreement in a way that improves the user experience and focuses on changing the approach for informed consent in EULAs. Finally, I demonstrate two models that show examples of practical implementation of these concepts and make some conclusions about their feasibility.
document. Typically, a check box is placed at the bottom of the window that the user is required to tick if they have read and understood the agreement,\(^1\) after which, if the response is positive, the software will install. An example is given in Figure 1.

![Figure 1: Microsoft Windows XP Service Pack 2 EULA screenshot](http://www.microsoft.com)

This disclosure approach makes no attempt to check to see if the user understands the agreement, often contains ‘hidden’ text (text deep within a long agreement that users would not generally read) that allows the software to perform tasks that may not be acceptable to the user, such as pop-up advertising, or which would violate the expectations of the user, such as privacy or security issues. These sorts of agreements are also prevalent in everyday use of computers, since each piece of software installed will generally require consent to an agreement similar to the one illustrated above. Typically, computers will have a number of pieces of software installed, which means that users have to consent to many such agreements. This process is repeated, often to the point where users know that it will be tedious to read all the material and skip through the stages without concentrating on the detail of the agreements. This ‘numbness’ is a serious problem that needs overcoming before we can claim to have a good informed consent procedure.

\(^1\) Whether or not they have read the agreement, ticking the box entitles the requester to assume it has been read and understood.
Thus, we have two sets of problems: one being the content of the agreement, and the other the display of the agreement (and acceptance mechanism). The content of the agreement, mostly legal text, is difficult to read, difficult to understand and, in some cases, almost deliberately obscure in order to dissuade users from reading it or to make it ambiguous in its meaning. The display of the agreement does not encourage a thorough reading of the agreement, even if it were easy to read, with the tiny text box holding thousands of words of text, often in all-capital letters, which can be difficult to read and comprehend. The consent mechanism in place (sometimes ticking a box, many times simply clicking ‘I agree’) is rudimentary at best, but coupled with the poor quality of the licence display, it is insufficient for the consent process to be considered informed. In order for this to improve, either the agreement mechanism needs to be changed (the more difficult route, since it could require some sort of randomness such as changing button locations to be introduced so that a new ‘numbness’ doesn’t eventuate), or the preceding communication framework (detailing the contents of the agreement) needs to be improved (a more attractive route, since it would more easily fit within the current paradigm, giving a much better cost-benefit ratio), or both.

The theory

The current theories used for informed consent in the medical and research fields comprise theories based on autonomy such as Faden and Beauchamp’s autonomous authorisation and effective consent models, or the general autonomy-based theory of Appelbaum et al (1987), yet both are insufficient (as is obvious from the current practice) for use in IT due to the high levels of expertise and face-to-face discussion interactions that need to take place in order for fully informed consent to take place. The current approach used for informed consent in IT is more closely aligned to the Faden and Beauchamp effective consent model than an autonomous authorisation approach or that of Appelbaum’s.

The effective consent model involves sets of policy rules that aim to regulate the consent-seeker in order to protect the consenter. It does this by utilising disclosure and competence tests that conform to a legal or generally accepted standard framework suitable for the context. It often uses appeals to autonomy as its justification, but requires little evaluation of any sort of autonomy on the part of the consenter (Faden & Beauchamp, 1986). This is particularly obvious in the case of EULAs which default to a de facto standard that was developed by the same set of people who also commonly act as the consent-seekers, or are closely aligned with other consent-seekers, and thus are more interested in providing the bare minimum for legally sufficient informed consent so that they are protected from liability. In doing this, they present a bare-boned
version of effective consent, which centres on disclosure yet provides almost no competence testing. This is an unacceptable approach for informed consent in EULAs, since it offers almost no attempt to inform the user of simple disclosure, which, as discussed, is presented in such a way that it is almost always ignored. This definitely benefits the consent-seeker over the consenter, and shows the need for an overhauled informed consent system for EULAs. On a theoretical level, it shows the need for rethinking the approach to informed consent in IT, since the situations found in IT are very different from those in the medical and research fields. I have therefore adopted Manson and O’Neill’s objections to the basis of informed consent on justifications of autonomy, and their revised theory that relies on the establishment of informed consent as a waiver of normative expectations.

Informed consent as waiver shows how the identification of normative expectations, coupled with a good communicative transaction framework, can improve informed consent procedures in computing situations. Choosing normative expectations needing to be waived restricts the scope of the informed consent procedure to the values considered important by the users of the software, rather than the more abstract and idealised values of the autonomy-based theories. A good communicative transaction framework allows for more direct disclosure of information that is relevant and accurate, rather than a general full disclosure that could easily flood the user with too much information, resulting in a confused or ignorant decision rather than an informed one.

The practical implementations and improvements discussed in this paper aim to not only provide an example for the application of the above theory, but also to demonstrate ways of overcoming developer and vendor apathy toward change. It does this by recommending a third-party mediation group that handles the development of new EULAs through an easy-to-use interactive interface. This group would provide a modular licence builder for simple and standard licence agreements waiving common normative expectations. These expectations would be standardised into language that can be easily understood, yet legally enforceable.

Some recent suggestions for improvements and practical implementations

The problem of informed consent in IT is not a new one. There have been several suggestions for improvements, including some user-tested implementations of these recommendations, such as with cookies in the Mozilla web browser (Friedman et al, 2000, 2002). Here I will look at some suggestions and implementations of solutions for the problem of EULAs, as well as some suggestions for how to encourage uptake and use of a revised system. These
examples were found through an extensive Internet search for suggestions for solutions to this problem (such as editorials calling for changes to EULAs), as well as actual solutions that have been developed (software, development frameworks), along with tools used for similar areas (such as copyright and fair use, certification for privacy or security) with the analysis concentrating on the benefits and limitations of each proposal or solution to the particular problem of informed consent in EULAs. Any proposal or solution that showed any sign of motivating improvements to EULAs (such as EULAlyzer) or informed consent decisions by users (such as Creative Commons, TRUSTe) was examined. Also discussed are the potential ways in which a trusted, third-party medium could enter the picture in order to attempt to regulate and standardise a system of licence agreements to make it easier for users to give more informed consent decisions.

The ‘Pure Software Labelling Act’

Simson Garfinkel compares the current situation with spyware and adware with the problems of patent medicines before the United States’ Pure Food and Drug Act of 1906 was passed (Garfinkel, 2004a). With the introduction of the Pure Food and Drug Act, ‘... with the knowledge of what they were to ingest, consumers were able to identify and avoid (if they wished) consuming potions that were “habit forming”’. Garfinkel claims that this could be a useful way to approach the problems of ‘overly- broad and turgid click-through licence agreements’ (Garfinkel, 2004a). Garfinkel specifically suggests using icons on a label at the top of licence agreements, and in other visible places, such as on install screens and in the Windows ‘Add or Remove Programs’ area of the operating system user interface. Software that displays such icons is not necessarily engaging in user-hostile practices, but the aim is to make it easy for consumers to easily identify which software displays which behaviour.

The icons he suggests are shown in Figure 2. They cover a few possible issues that users encounter in software, and which could be considered violations of normative expectations.

Figure 2: Label icon suggestions by Garfinkel (2004a) for software licence agreements

Source: Garfinkle, 2004a
One of these issues is software that modifies the operating system, usually to access pieces of hardware by installing drivers, or to access some of the lower-level functions of the operating system, such as manipulating the system memory used by other applications. Another is the issue of monitoring, such as in the example of keystroke logging, where a piece of software keeps a record of every keystroke made on the keyboard (this is often used by malicious software aiming to access usernames and passwords to websites and other services and information contained in private files), or software that tracks what websites the user visits from their web browser. Pop-up windows are also addressed, as is software that downloads updates for itself ‘that could change its behaviour’. Although diallers have become outdated, since many Internet connections are no longer directly through a phone line, they used to be a major issue, because the software would hijack the phone line and make expensive phone calls (often overseas!) without the knowledge of the user. Another issue Garfinkel recommends highlighting is the problem of software running at start-up. Software that loads its core functionality when the computer starts up appears to load faster when the application icon is later clicked by the user, but, ultimately, it makes the computer run slower, even when the application is not being used. When many applications use this functionality, the slowdown becomes noticeable, especially when the computer starts up. Also, software that allows others to control the computer remotely (I believe the intention of the author is to call that label ‘Remote’ rather than ‘Remove’), such as with the popular software program called VNC, often used in company technical support situations, and software that does not allow itself to be easily uninstalled would get their own label under Garfinkel’s scheme.

One requirement of Garfinkel’s idea is that the labelling be mandatory and legislated by government. In this he appeals to the Pure Food and Drug Act’s success in giving people the knowledge of a food product’s ingredients in order to effect changes in peoples’ decisions regarding their consumption of food items. Such mandatory labelling would also cause companies to change the inclusions in their products, in response to public acceptance or rejection of those inclusions. At any rate, Garfinkel’s idea is worth considering, given not only the success of the Pure Food and Drug Act, but the appealing nature of labelling things simply.

The benefit of this sort of system lie in the fact that it would be immediately obvious what norms would potentially be violated should the user install the software. A standard set of icons for the label would be available for software manufacturers to describe the activities of their software, and an external group would have jurisdiction over the use (and misuse) of the labels. It would have the potential, if fully enacted as Garfinkel proposes, to change the behaviour of software manufacturers, that is, some mechanisms would no longer be used because they would be considered detrimental if obviously part of the software
(such as pop-up advertising or hijacking the telephone line). Overall, users would have significantly more control over their software through market force, and would appreciate the easy way to determine what sort of behaviour is intrinsic to their software.

Of course, there are (somewhat unfortunately) obvious problems with this approach. For one, unlike the importation and buying and selling of food and drugs, software is not physically restricted within a particular jurisdiction. The availability of software on the Internet makes it difficult to require mandatory labelling, since software companies could easily set up in a location that doesn’t require this standard, especially companies that rely on expectation violations to make money, such as advertising software companies. The market forces at work here mean that there is a lot of momentum required to change procedures, since markets only improve products under competitive conditions. This doesn’t make the idea unviable, however, because ideally the companies that work out of that jurisdiction would be able to establish that use of the labelling system would have advertising, and therefore revenue, merit as well as inherent moral worth. This sort of marketing strategy could improve both the company brand and the view of the licensing scheme, with a labelling culture eventually seen as a competitive status symbol.

Garfinkel points out another issue, in that it would be difficult to decide what sorts of potential expectation violations should be labelled.

The more information required on the label, the more expensive it will be to produce, and the less likely that consumers would be to actually pay attention to the information. Any regulatory body implementing this policy will need to avoid icon creep — having 23 different icons on each piece of software won’t serve the needs of consumers, it will just cause confusion. (Garfinkel, 2004b)

This is an important point, and one that would have to be dealt with comprehensively, since it is clear that there would not only be potential creep but also the requirement to deal with old and outdated labels as well (such as in the case of the dialler).

This could be overcome, however, by developing icons that can encapsulate at least the essence of the issue, thereby allowing users to acquire a reasonable idea of the software contents. This would still be preferable to the computer user having to wade through 5000 words of legalese, as is the current expectation.

Overall, Garfinkel’s proposal is an interesting approach to solving the problem of difficult-to-read licence agreements. Unfortunately, it is unlikely to work in its current incarnation, since there is little momentum behind it (little active advocacy or any real attempt to institutionalise it) and it requires government
legislation to be fully effective, which requires lobbying and probable industry backing, or other critical mass elements. It also ideally requires global acceptance and institution in order to become the dominant culture in the Internet-enabled market. This problem, as well as some of the other limitations discussed above shows that, although this idea does not fully solve the informed consent problem, it certainly is a step in the right direction.

StopBadware

StopBadware is a database of known ‘badware’ (such as Trojan horses,2 or viruses that exploit web browser vulnerabilities) and badware-serving websites. Some large companies and popular software, such as Google and Mozilla Firefox, use this service to help protect their users against malicious software.

StopBadware is not specifically aimed at improving informed consent, but it provides a service that shows how dissemination of information about the importance of informed consent in IT situations can be achieved and, ultimately, identify improvements to the quality of software to help to improve informed consent. StopBadware’s mission statement declares that they are ‘... committed to protecting Internet and computer users from the threats to privacy and security that are caused by bad software’ (StopBadware, 2008b). They do this by maintaining a database of badware, that is, software that violates their guidelines for good software design. In general, their definition of badware is this:

An application is badware in one of two cases:

1. If the application acts deceptively or irreversibly.

2. If the application engages in potentially objectionable behaviour without:

   • First, prominently disclosing to the user that it will engage in such behaviour, in clear and non-technical language, and

   • Then, obtaining the user’s affirmative consent to that aspect of the application.’

(StopBadware, 2008b)

There are specific references to informed consent within the guidelines, with StopBadware requiring applications to disclose the installation, origin, function, and any potential ‘unexpected or unwelcome’ behaviour of the software, and then seek consent for these. This, indeed, is a very specific example of Faden

---

2 Malicious software that ‘hides’ inside legitimate-looking software, installing invisibly when the legitimate-looking software is installed. The ‘rootkit’ is also a type of Trojan horse.
and Beauchamp’s effective consent model, that is, the requirement of disclosure, and then the consent to the terms disclosed (although there are no tests for understanding on the user’s behalf). It is with this in mind that I discuss this example because, although it has a large amount of focus on informed consent, it doesn’t actually take any steps to improve the current flawed consent model. Instead, the StopBadware model simply requires an EULA and privacy policy to be present and ‘written in as understandable a manner as possible’.

Thus, I use this example as a general illustration of how a better model of informed consent could be instituted: the StopBadware database of badware is highly regarded and successful, and the organisation has some powerful industry backing, with the likes of Google, Mozilla, PayPal, AOL, VeriSign, and Trend Micro among its partners. The reason it is so successful is because of its integration into Google and Mozilla Firefox. By accessing Google, the most popular Internet search engine, and Mozilla Firefox, one of the more popular web browsers, many users encounter the work of StopBadware. The StopBadware process works as follows: if a website is reported to StopBadware as distributing (knowingly, or unknowingly due to being hacked, for example) badware, its details are entered in a database which is checked by Google before it displays Internet search results. Google then displays a warning to the user before the user clicks through to the reported website, stating that ‘this site may harm your computer’ (Google, 2008). Firefox will also display a warning before loading the website (StopBadware, 2008a). Because companies rely on Google placement in search results, the detrimental effect of having a warning placed on their website impacts their business, and thus effects change in their behaviour. StopBadware has a ‘clearinghouse’ of many software applications and websites that have improved as a result of this (StopBadware, 2008b).

This shows that there is certainly a way to encourage software vendors and manufacturers to change their behaviour without legislative requirements or the need for a critical mass. Instead, social pressure can be placed on companies through negative publicity, with companies eager to be determined suitable for exclusion from the badware list. StopBadware, Google, and Firefox thus have a significant ability to effect change and, although they mostly target dangerous software, they also require that software display certain positive behaviours through disclosure of information through an EULA and/or a privacy policy. This could potentially go further by requiring certain ways of displaying agreements and policies, or including a labelling scheme similar to Garfinkel’s. It should be noted, however, that excessive regulation could be detrimental to the project, or that this could be seen as an abuse of power (to negative effect) or as a ‘slippery slope’ to other requirements for warning-free display on Google or Mozilla Firefox. So long as it is simply a warning and not a complete restriction mechanism, the potential good should outweigh the slight inconvenience for non-conforming software websites. Overall, it is a considerably effective
delivery mechanism for protection of users from potentially dangerous software on the Internet, and could be used for further improvement of informed consent procedures beyond immediate jurisdictions.

**EULAlyzer**

EULAlyzer (Javacool Software, 2007) is a piece of concept software that takes any EULA and analyses it for particular words and phrases considered ‘interesting’. These include words and phrases involving advertising, third parties, changes that may be made without notifying the user, or privacy information. When a user installs a new piece of software, they can use the EULAlyzer software to copy an EULA into the analysis tool, which EULAlyzer scans for terms of interest, displaying the results in a window for the user to easily scan through the pertinent phrases and work out if they are important. A screenshot of its operation can be seen in Figure 3. In this figure, the software licence agreement for Download Accelerator Plus (SpeedBit, 2008) has been analysed. In the upper screen, a list of suspect text is shown (‘Flagged Text’); this shows the word as well as the context of the word as it appears in the EULA itself. The user can then click on the green arrow beside it to go straight to the pertinent spot in the EULA on the bottom half of the screen, where they can read more of the surrounding text. The ‘interest level’ shows how likely it is that the user would be interested in those pieces of text. I found that gauge unhelpful, since I was interested in seeing what all of the flagged text was, rather than selectively looking at or ignoring other phrases. The user interface screen for choosing the EULA to read was a difficult to use, since the window of the EULAlyzer couldn’t be resized easily and the program requires the user to drag an icon from the EULAlyzer program to the text box of the licence agreement on the installation screen of the software in question.

The difficulty of dragging the icon and completing the procedure is likely to be discouraging for users of the free version of the program. Since the point of such a program would, ideally, be to encourage use and therefore improve understanding and knowledge of the contents of EULAs, the free version of EULAlyzer is not the best solution to the problem of EULAs.

There is a more advanced version of the software (which requires payment; the simpler version is free) that runs continuously in the background, and analyses any licences that the user encounters when installing software. This is much easier to use, since the user doesn’t have to open the program to run the EULA through it each time that they install software (which would mean they could easily become lazy and not bother to actually do it). It also only runs on one platform (Microsoft Windows) and this limits its usefulness. It would be beneficial if this, or a similar program, became standard on all operating system platforms, but, as it is, it is simply a niche market application that appeals to the more privacy and security-conscious computer users.
EULAlyzer is essentially a patch approach to the systemic problem of informed consent in EULAs and, although it is useful, it doesn’t attempt to change the poor behaviour of companies, but simply provides an option for users who are concerned about their computer and data’s welfare. Since it is also analysing the EULA documents by scanning for certain words, it could also ultimately be a reasonable expectation that software wishing to avoid clauses being flagged by the analysis could rewrite their EULA to use words that do not trigger an alert in the software. EULAlyzer would then give a false sense of security, and could ultimately be more dangerous than not having the analysis tool at all.

TRUSTe

TRUSTe is an independent body which provides a service certifying that a website or other Internet service (including email lists), or software conforms to their best practices with regard to privacy. I am interested in the ‘Trusted Download Program’, since TRUSTe makes the claim that, with their certification, the software cleared by the program has been proved to be spyware free, and that they can ‘build [their] reputation and increase consumer confidence’, ‘open doors to major advertisers and partners’, ‘reassure distribution partners
and increase downloads’, and ‘join the only industry-recognised safe-software whitelists’ (TRUSTe, 2008b). TRUSTe go on to state that the software certified by them ‘directs software distribution opportunities and advertising dollars to programs and affiliate networks that respect consumer choice’. Essentially, it is a middleman between software companies and advertising companies, with a degree of respectability brought on by the standards that they require of the software, which provides a certain exclusivity. The requirements of software for compliance with the program are for ‘transparent distribution practices’, ‘clear disclosure and informed consent prior to download’, ‘clean uninstall’ and ‘respect for the end-user’ (TRUSTe, 2008b).

The organisation has close ties to advertising companies and download websites that subsist on advertisements (through both adware and website advertising) for their income, which raises questions as to their integrity. They do, however, have a very comprehensive list of specific policies that the software must abide by (and be subject to audits on), and the penalties are reasonable (consisting of being removed from the program, possibly with public advertisement). They also have strict requirements for EULAs and consent, including that on installation, the software displays a primary notice of functionality that could impact the user, and that an EULA (or opt-out mechanism) is ‘insufficient for providing such notice or obtaining consent’ (TRUSTe, 2008b). These are, for the most part, excellent policies, however, looking at the published ‘whitelist’ (TRUSTe, 2008a), there are a large number of questionable pieces of software, especially the advertising and tracking software listed, which includes some software with particularly questionable activities. One such example is the software WhenU Save/SaveNow, which is listed as generally unwanted adware by many anti-malware programs (Healan, 2005), and which is often packaged as a third-party piece of software along with other ‘free’ software, such as the aforementioned screensavers or movie players. Perhaps TRUSTe’s for-profit status, and reliance on companies to pay subscription fees to display the certification seals, means that it needs to form working relationships with these companies in order to work out any potential difficulties rather than simply offering the service as a simple ‘tick or cross’ system. According to Hansell (2008), TRUSTe has a record of questionable toughness on its clients:

… it does not always tell the public if it discovers violations of its principles, even if the violations are so egregious that it kicks a site out of its program. Last year, it ejected three of the 1500 companies in its certification program, and three more chose not to continue because their business models no longer complied … The organisation declined […] to name those six companies, saying it only makes public cases of ‘blatant violations with probable consumer harm’. 
It also encountered image problems with its approval seal for certified websites being targeted in cross-site scripting attacks, which allowed spamming websites to illegally display the image in order to lull the consumer into false confidence about its activities. Worse, these attacks tricked the TRUSTe server into confirming these websites as legitimate, thus misleadingly approving them. This was more destructive than a simple re-using of the seal image to instil a false sense of security in users, because it returned results that said the use of the seal was legitimate (Wagstaff, 2004). This problem undermined the system for TRUSTe and, although the issue was resolved, it was damaging to their image as a trusted third party. A similar issue occurred when the lists of TRUSTe-certified websites was correlated with McAfee’s SiteAdvisor, a service that tracks websites that have viruses or other problematic exploits, and there was a number of TRUSTe certified sites on the SiteAdvisor list. This, too, was problematic for TRUSTe’s image, since its aim is to certify a website as free of these sorts of exploits and viruses (Leyden, 2006). While these problems are not unique to a service like TRUSTe, the fact that their auditing process was flawed is of note, since it would seem obvious that a website operator could set up a site that conforms to the TRUSTe guidelines then, using the respectability that comes with the TRUSTe label, gain trust of its own. The operator could then start to take advantage of that trust by introducing viruses or exploits on the site, taking as much advantage of it as possible before TRUSTe investigated the breach of its policies.

This sort of approach is attractive to software companies that are engaged in creating and promoting software that requires more credibility than usual: makers of advertising software or software that includes some other feature which may be considered unwanted or violating major expectations of a user, such as a keylogger or a client for a remote access server. These sorts of applications already have a certain stigma associated with them; recognition by a service such as TRUSTe is one way to gain respectability. Thus, it seems that it is unlikely that this sort of approach will eventually encourage a critical mass of software manufacturers, since many, like Google, like to at least claim to ‘keep good company’ (Google, Inc., 2006), and not taint their brand with association with these companies and their software.

This is not to say that all third-party certification is a bad thing: becoming certified as conforming to ISO standards or Hazard analysis and critical control points HACCP standards for food safety, for example, is widely regarded amongst the relevant industries. TRUSTe, unfortunately, does not have the reputation of these standards organisations. Ultimately, a reputable, third-party standards organisation can enhance the reputation of software, but, until standards are developed that are well-balanced and focus on identifying values and setting up frameworks for dealing with user and company expectations, this remains
a problematic process. The guidelines discussed here could be used to develop such a framework for independent, third-party standards assessment, and could be used as the ethical basis for a legal regulatory framework that would support such a standards organisation.

Creative Commons

Creative Commons is a non-profit organisation that provides licences for use of creative works; licences that span a variety of different restrictions and freedoms in an easy-to-access model allowing users to easily licence their work. Their work is based on the idea of a ‘commons’, ‘resources that are not divided into individual bits of property but rather are jointly held so that anyone may use them without special permission. Think of public streets, parks, waterways, outer space, and creative works in the public domain’ (Creative Commons, 2008b). Thus the licences allow the holder to retain their copyright, but allow people to distribute, modify, and/or copy the work, providing that due credit is given (depending on the specifications chosen by the user).

The website allows users to choose a licence after answering a series of questions, such as whether to allow commercial uses or modifications of the work, and the jurisdiction of the licence, and then presents it simply, as is illustrated in Figure 4.

Note the use of icons, similar to the type Garfinkel proposes, to allow for an easier understanding and a more immediate recognition of the requirements of the licence. The standardisation of the licences, too, makes it easy for people wanting to use the work to know what requirements they need to fulfil. The third-party nature of the Creative Commons organisation means that it is not simply acting on the copyright holder’s behalf, but has sought to make the licences and licence procedures understandable and fair to the user.

Another positive side to the Creative Commons approach is the easy-to-use form. Within a few minutes, a copyright holder can have a socially acceptable licence drawn up that has a solid legal foundation. This independent, non-profit, unbiased third party runs from donations and is part of a grassroots advocacy organisation with a large support network (Creative Commons, 2008a). In this way it has attracted many of the newer websites, particularly social networking and media sites such as Flickr, Google, Yahoo! (Creative Commons, 2008b) and, with them, many users of the Creative Commons licensing system. With this large industry backing, Creative Commons is a good case study for reasonable ways to approach change in culture within the IT industry.

Although there is the potential for misuse of the Creative Commons labels, similar to the issue face by TRUSTe, because the use is within copyright and contract law, misuse could be opposed by legal action. The situation is different with TRUSTe, which has no legal backing beyond striking a certificate from
its list. What Creative Commons does differently is not simply act as a third-party auditor, but provide a service with which users and companies can create licences with which to claim their own legal rights.

It is with this in mind that I move on to my recommendations for practical implementation of informed consent reform within EULAs.

**Figure 4: An example of a Creative Commons license, with simple icons and explanations: The licence illustrated here is the Attribution–Noncommercial–Share Alike 2.5 Australia licence**

Source: Creative Commons, 2008

**Recommendations for practical reform of EULA informed consent**

Having analysed the current approaches to dealing with informed consent issues in IT, it is clear that there are several different ways of, firstly, appealing to the industry and, secondly, making these consent issues meaningful for
both users and manufacturers, thereby improving the overall level of consent. TRUSTe and Creative Commons are (currently) both non-profit organisations, yet one attracts a particular market (software requiring certification in order to gain a certain amount of respectability), whereas the other has a much wider base, with a highly respected group of industry links that use their system, and considerable respectability attached to it. The difference between them is that one requires end-user trust in the label and protection of that label, whereas the other requires trust in the system in order to obtain a legally backed and protected label. It is obvious, also, that a common suggestion for improvement of ease-of-use for users of such systems is by employing icons to allow for easier identification of important information. Also important are ways for industry to incorporate the system into their own systems, such as StopBadware and Google’s approach to improving consumer awareness of potentially dangerous websites. It is obvious, too, that there needs to be a more basic approach than the EULALyzer, which works to reform the system rather than simply deal with it as it is.

In this way I propose an EULA system by which a genuinely independent, third-party, non-profit organisation establishes a set of standard agreements, in a similar way to the standard licences from Creative Commons, coupled with an iconic labelling system that identifies key aspects of the agreements that users would consider important, in a way that allows users to waive normative expectations in order to give proper informed consent.

In establishing this system, I will outline some basic normative expectations that I would expect to be the main focus of such a system, and reiterate the process by which to identify the expectations, discuss the merits and potential issues with a set of standard agreements, along with a process for adding additional clauses to a standard agreement under this system, and then discuss the method by which such licences would be distributed, through a Creative Commons-like, non-profit trusted mediation service.

**Choosing normative expectations**

The expectations users have of software vary considerably, but there are several important contraventions of these expectations that can easily be outlined as being potentially harmful to the user or having a negative impact on the performance of the computer, or are generally actively avoided by users, such as advertising.

Some potential activities of software that would generally violate a user’s normative expectations include (but are not limited to) the following:

- monitoring of user behaviour
• collection of personal information
• sending personal information to a third party
• changing the behaviour of other software
• changing the behaviour of the operating system
• installation of additional software
• display of advertisements
• requiring extra computing skill to remove
• starting automatically on boot
• running in the background
• automatic upgrading.

A user would need to know if their behaviour is being monitored, such as through software that monitors keystrokes or captures screenshots, or which monitors traffic flow over the network. Display of advertisements is another problematic function. Other activities that affect the user could be the practice of directing the software to start up when the computer first boots up, and then running it in the background seemingly invisibly until the user actually wants to use the application, at which point it loads very quickly because it doesn't need to perform all of its start up routines. This is not a bad thing on its own, and can result in speedier loading times for programs. The problem is that with many programs performing this activity, the computer starts to take a long time to start up on boot. This could mean that the computer actually starts to perform much worse than usual because of all of the programs that are running in the background, taking up system resources such as memory and CPU cycles and thus encumbering the computer. Automatic upgrading is also a potential issue, since, although it is usually welcomed by most users and recommended for security issues, there are some upgrades that could be harmful to particular users, especially if the upgrade significantly changes the behaviour of the software. These sorts of changes could be problematic for users that rely on a particular feature of the software that is removed by the upgrade, so there should be at least some sort of notification and re-acceptance of the agreement before the upgrade is made.

Other things not directly related to software activity that should be considered for explicit waiving within an EULA include the normative expectations of merchantability and fitness for a particular purpose, complete termination and deletion of details of any account information on uninstallation or unsubscription, the ability to obtain a refund if the terms are not agreed to, and notification of changes to the terms of the agreement.

Sets of logos or icons would need to be developed in order to best represent the ideas for waiving. The sorts of things Garfinkel had for his set is a good
start, but more would need to be developed. The problem of an excessive number of meaningless icons being displayed could be avoided in several ways. Accurate and representative graphics that respond with a mouse-over display summarising the icon’s meaning would be a useful starting point.

**Standard agreements**

With the normative expectations identified, and mechanisms in place to allow users to confidently waive them with informed consent, it is important to look at the other end of the system, that is, the rights and responsibilities of the software writers and vendors, which were previously the sole focus of an EULA. To effectively deal with the legal aspects of EULAs, I propose that a set of standard agreements be drawn up, with the basic legal requirements available to be added in as ‘modules’, similar to the drop-down menu selection screens of the Creative Commons website (Creative Commons, 2008b). In this way, licencers could choose which normative expectations licencees need to waive in their agreement, with appropriate text and icons inserted into the created agreement. It would, by necessity, be a more complicated than the Creative Commons example, but, considering there would be fewer licences compiled and there are greater ramifications of poor informed consent at stake, this would be a necessary and reasonable trade-off to make.

![Figure 5: The Creative Commons licence choice screen](source: Creative Commons, 2008)

The use of modular standard agreements would be a convenient, clear, and effective way to communicate rights and responsibilities of both the user and consent-seeker, and to obtain waivers of a user’s normative expectations in order to install and use the software. Not only would a user be able to predict the content of the agreement by the icons used to draw attention to potentially
problematic areas of the agreement, but there would be a greater level of trust in there being a minimum of the previously identified ‘hidden’ or otherwise ambiguous clauses.

Should any conditions required by the licencer fall outside the provided sets of modules, a separate ‘additional clauses’ section could be provided, on a second screen, stating that they fall outside the conditions provided by the standard licence template. Generally speaking, these instances should be discouraged, or be considered for module status with appropriate standard wording devised, but, for more obscure software inclusions, this option should be available.

**Trusted mediation**

With these sorts of requirements for improvement of informed consent in EULAs, it would be necessary for an organisation, similar to that of the Creative Commons, that develops, maintains, and lobbies for the use of the above system. In particular, the organisation should be non-partisan, resulting in a legally supported and industry backed mediation group that is trusted by both users and industry to be working in the interest of all stakeholders.

Such an organisation would instil confidence in the users of software, in line with John Weckert’s (2005) discussion of solutions to the problems with the establishment of online trust. In this, Weckert notes the use of certificatory third-party endorsements by way of ‘seals’, for which a company must satisfy certain requirements before they can display the seal (Weckert, 2005). Unlike the TRUSTe attempt, however, there would be no requirement for membership or licensing fees, thus eliminating the potential to favour one set of stakeholders over another. The lack of accountability to anyone but the organisation would only serve to strengthen the trust placed in the organisation by all stakeholders, since it would act as a mediation between groups with different sets of moral values. Also, since the licence is incorporated into the installation procedure of the software, there would be little room to doubt the authenticity of the licence, as it would have to read and perform the same way as the sample licences, or be held unconscionable if brought to legal action since outright deception would be taking place.

Thus, setting up a trusted mediation organisation that is non-profit and concerned with all stakeholders is the best way to develop, maintain, and lobby for these improvements to EULA policies.
Theory in practice: An example

The following example of an EULA, which puts the theory into practice, would require both legal and user interface design expertise for commercial application. The system that I propose assumes the oversight of a third party that is trusted to establish and govern the standard agreements. The situation that I outline could, however, be used outside of a third party too, given the consent-seeker is able to derive their own normative expectations from the guidelines and recommendations outlined previously.

Software background and purpose

The software in question, which is entirely fictional, will be named Admo. Admo is software that allows content providers, such as video or game developers/ producers to generate an income from their work by way of advertisement support. For a computer user to view such videos or play games in this scheme, they will need to firstly install Admo, which will present them with targeted advertisements once an hour by way of a popup window. The advertising software will use information gathered from the computer user’s web-browsing habits to work out what advertisements would be appropriate.

Standard agreement and modules

The trusted third party would ideally provide a basic standard agreement, which outlines the requirement for the user to accept the agreement before use, identify the jurisdiction under which the licence is legal, and other legal requirements that do not involve the software activity or function specifically (the exception to this is the module requirement for notification of policy changes). This section would incorporate the traditional first few paragraphs of EULAs. The standard agreement provided by the trusted third party would include modules for the waiving of the selected expectations, chosen for a particular jurisdiction. Each module should be assessed for inclusion depending on the software’s requirements and purpose. The modules in italics below are the modules that Admo will need to seek explicit consent for waiving. All expectations it adheres to would be listed in positive language within the body of the agreement as well.

These modules include, for the purposes of this example:

- The software meets minimum standards of merchantability and fitness for the purpose for which it is made (according to law in that jurisdiction).
- The software does not disrupt the existing operation of the computer.
• The software does not distract the computer user.
• The software does not track the behaviour of the user when using the software.
• The software does not track the behaviour of the user when using other software.
• The software does not collect personal information for its own use.
• The software does not collect personal information to sell or give to third parties.
• The software allows a user to easily opt-out or uninstall, that decision will be final and persistent, and result in the removal of all traces of the software and the user being unsubscribed from the service.
• The software does not install additional software.
• The user may return the software to the place of purchase, unused, for a full refund if they do not agree to the terms of the agreement.
• The user will be notified of policy or specification changes and need to give explicit consent to these changes.

These modules (of the software Admo) would be developed by the trusted third party (not the software company!) from normative expectations about the particular type of software (downloaded from the Internet, requiring some Internet connectivity to function, as opposed to being bought from a shop and not requiring any Internet connectivity to function), based on industry guidelines and focus groups (or similar). They would have accompanying legally binding text, such as is seen in current agreements. Admo’s company simply needs to decide which ones are applicable to their software, which will be highlighted and indicated plainly on the main window of the agreement.

**Communication: EULA design**

Once Admo has a set of normative expectations to waive associated with it, an EULA window for their software will be created to facilitate the communication transaction that needs to take place to fulfil the requirements of the theory. I outline here two possibilities for the design of the agreement panel, which would run on installation or re-request for consent. One is a two-window design, and the other displays multiple windows depending on how many modules are used (that is, how many normative expectations need to be waived). It could be that these are organised hierarchically, or kept separate, but determining this and which sort of model is suitable for the target audience is best left to further feasibility studies and trials.

The major problem with designing agreements is the need to balance the requirement for informed consent with an annoyance factor on the part of the user at having to perform several minor tasks before installation of the chosen software is complete. The more detailed the consent process, the more likely it
is to irritate the user, which is why I propose two slightly different models, one of which is not as involved as the other. Ultimately, though, I suspect that the overwhelming desire to install the software would override the annoyance of the EULA windows, especially if there were obvious benefits to the requirement of such windows.

**Two-step agreement**

This model would use a similar style of screen (or window) to the current generic EULA window, but the user would have more control over the process. While the user could access the legal document in its entirety if they wish, the icon highlights would act as signifiers to the user of items for potential concern within the agreement. Hovering the mouse over the icon would bring up a tooltip with a short explanation of the behaviour requiring explicit consent. Clicking on the icon would take the user to the pertinent point in the formal document, which would be written in plain, yet legally binding, English (or have a plain English equivalent), and highlight the area of interest. A requirement to check off each expectation, as encapsulated by its icon, could be built into the screen. If one or more of the individual expectations are not waived by the user (that is, the checkbox under the icon is not checked), some more information about each one could be communicated to ensure that the omission is intentional, with an explanation that it will mean the software will not be installed. The agreement also asks for the user’s date of birth and location, as a basic form of competence testing and jurisdiction identification. A mock up of the first example window for this style of agreement is displayed in Figure 6.

To address expectations needing to be waived but not present in the collection of modules, a second step can be optionally added which allows for additional expectation waiving. Here, any expectation that does not fit under the previously waived expectations should be listed. An example of the second step window is in Figure 7. This allows for an element of futureproofing and flexibility for the scheme. The danger is, however, that with technology changing, new types of expectations would start to become apparent, and this second window could become cluttered. It is because of this that the trusted third party should monitor the sorts of expectations that are outside the standard ones, and eventually include them in the list of standard expectations, while retiring those that are no longer necessary. This style of consent window, though, means that consent-seekers, who might otherwise only see the standard modules as being required for disclosure and consent, cannot modify the standard agreement to ‘hide’ other things in it, and so must have a visible second window that draws attention to additional requirements.
8. Informed consent in information technology: Improving end user licence agreements

Figure 6: Two-step agreement, first window, showing main standard information and modules

Source: Author’s research

Figure 7: Two-step agreement, second step, showing extras

Source: Author’s research
Multiple-step agreement

The second option I present is a more thorough attempt to inform the user before seeking consent. It starts with the standard basic agreement and then moves on to a single window per expectation module (or group of expectations).

Figure 8 illustrates a similar window to Figure 6, except it presents only the basic agreement. The same basic competence test is set out (asking for date of birth and jurisdiction), and the language is simple. Accepting this window of the agreement would bring the user to the next step in the agreement.

![Figure 8: Multiple-step agreement, first window, showing standard information (no modules)](image)

Source: Author’s research

The following step is where the bulk of the expectation waiving is done. Each expectation module, possibly encapsulating related expectations, shows a set of icons identifying the expectations, and a summary of the effect of waiving those expectations (shown in Figure 9). Below the summary, the effects are explained in full, with a checkbox for the user to agree to each section, and identification
of how many steps remain in the process. After agreeing to a module, the next module appears, until all modules have been agreed to. In this way it is much easier for the user to see more information about the potential issues that might affect them. Finally, following the module acceptance, an extras window, as shown in Figure 7, would be presented, with the same potential benefits and drawbacks as discussed in the two-step agreement, above. Only after accepting every step in the agreement would the installation of the software continue.

Figure 9: Multiple-step agreement, second window, showing information for each module

Source: Author’s research

It is likely that the uncluttered appearance of the multiple-step example will meet with more approval. While it gives the user more useful information in easier to understand short statements, it is more likely to cause annoyance because of the number of windows requiring the attention of the user. The annoyance factor may be mitigated by the education of users in the importance of giving informed consent to EULAs.
Conclusion

In this paper I have outlined my recommendations for improvement of informed consent in IT by improving the worst offending area: that of EULAs. In doing this, I outlined the obvious problems plaguing EULAs, and then outlined some of the suggestions that have been put forward to improve the situation. Some of the suggestions were directly related, such as Garfinkel’s labelling system, whereas others provided examples of trusted third party support mechanisms such as the Creative Commons and TRUSTe. I evaluated each one in terms of the problems of EULAs, and then established my own recommendations: a hybrid system of labelling and trusted third-party mediation and support that provided a foundation and design paradigm for modular standard license agreements. In keeping with the normative theory I have adopted for this task, I identified the normative expectations, discussing them in the context of the suggestions made in the previous chapter, and then established an effective communication framework: using trusted third party mediation to develop a set of modular standard agreements that are easy for users to understand, with labels identifying the particular normative expectations that would need to be waived for each piece of software. I discussed the merits and problems of the system, particularly some places that would need workable trade-offs, such as the potential problems of crowding of labels on the license window, lack of label recognition, and complication of license choosing. I also dismissed a few problems that have plagued similar projects, namely the problem with ‘seals’ in that they are easy to forge. Finally, I demonstrated a practical implementation of the theory proposed for the case study of EULAs. I offered two models, each with benefits and disadvantages, and suggested some further research for trialling these models.

In doing so I have outlined a serious practical application recommendation for improvement of an example of informed consent in information technology.

References


Creative Commons, 2008, website, viewed 20 October 2009, <http://creativecommons.org>

Friedman, B, Millett, L & Felten, E, 2000, ‘Informed consent online: a conceptual model and design principles’, Tech. rept., 00-12-2. UW-CSE.


