Bug Tracking Systems as Public Spheres[1]

Joseph Reagle, Jr.
Department of Culture and Communication
New York University
The Steinhardt School of Education

Abstract
Based upon literature that argues technology, and even simple classification systems, embody cultural values, I ask if software bug tracking systems are similarly value laden. I make use of discourse within and around Web browser software development to identify specific discursive values, adopted from Ferree et al.'s "normative criteria for the public sphere," and conclude by arguing that such systems mediate community concerns and are subject to contested interpretations by their users.

1. Introduction

> "Last time I filed a bug report with KDE I got some snotty reply from some programmer who said I was wrong ([even so] the bug got fixed in the next release and was listed in the changelog)". - ErichTheWebGuy
>
> "I've submitted a number of bug reports and comments on existing bugs, and not only were they fixed promptly, but my privileges were raised so that I could close bug reports/mark duplicates/etc." - Anonymous Coward

The two comments above (Michael 2004) represent opposing positions within a discussion about reporting software bugs. A "bug tracking" tool permits one to identify, discuss, prioritize, close, and remove duplicate reports of system deficiencies. When most people think of bug and issue tracking software, if they do at all, they would probably think that it is a peripheral and mundane technology. Yet there are complex technical and social processes involved in addressing software bugs (Bork 2003).

As indicated by the frustration of "ErichTheWebGuy" above, a source of disagreement and even exit within open source communities is the handling of software bugs. In the open standards and software communities that this paper considers, the ways in which issues are represented with respect to their standing of consensus or dissension is affected by the processes, culture, and media of discourse (e.g. IRC, e-mail, Wiki, etc.).

Consequently, this paper is an exploration of how issue and bug tracking tools "embed," "embody," or "inscribe" cultural values of how a community should come to agreement, or even productively disagree. For example, what categories are available to describe the closure of a contentious issue? Or, how are the resource costs of reporting versus resolving a bug balanced?

2. Background: Values, Bugs, and Discourse

[1] Acknowledgment: I would like to thank Helen Nissenbaum for her comments on and discussion of this paper, and Nora Schaddelee for reviewing an earlier draft.

2.1 Values Embodied in Technology

In her provocatively entitled paper *Do Categories Have Politics?,* Lucy Suchman (1998) attacks the theory behind a Computer Supported Cooperative Work (CSCW) tool known as the "Coordinator." The operation of this tool was predicated on a foundation known as "speech actor theory." Suchman faults Winograd and Flores, the proponents of this theory and the designers of this tool because "the adoption of speech act theory as a foundation for system design, with its emphasis on the encoding of speakers' intentions into explicit categories, carries with it an agenda of discipline and control over organization members' actions" (Suchman 1998).

Terry Winograd responds to Suchman's question of "Why do computer scientists go about making up all these typologies of interaction?" (Suchman 1998) with this pragmatic reply: "The answer is relatively simple -- computer programs that we know how to construct only work with fully-rationalized typologies (be they bits and bytes or knowledge bases)" (Winograd 1998:109). Winograd acknowledges the potential problems of this process and notes: "The essence of using a tool well is knowing where, when, and how to apply it" (p. 111). This is reminiscent of the argument that guns don't kill people, people do. And while this essay will not address this complex question of a designer's responsibility – regardless of their intent – to all potential applications of their artifact, Winograd (p. 111-112) does offer some qualifications with respect to this type of design:

1. Explicit representation of intentions and commitments is more appropriate in some social/organizational situations than others.
2. The generation of representations can only be done successfully with the participation of the people who live the situations being represented.
3. It is a dangerous form of blindness to believe that any representation captures what is meaningful to people in a situation.

Yet, each one of these caveats merits a substantive discussion as well. Unfortunately, at this point I must avoid the particulars of that discussion to focus on what I hope the reader will accept as a presumption: that -- as Langdon Winner (1986) might say – "artifacts have politics." What are politics? Winner defines them as "arrangements of power and authority in human associations as well as the activities that take place within those arrangements" (p. 30). Not all politics are equally political, or political in the same way. In any case, my analysis is predicated on the simple point: technology is created and used by humans, and in both of those acts the technology interacts with and mediates the human/social sphere.

By way of example, Winner notes the wide Parisian thoroughfares that were intended to mitigate revolutionary barricades, the American university campuses built to facilitate easy troop movement and sniper positions during students protests, and the deployment of less efficient machines to displace unionized labor. Pinch and Bijker (1992) use the development of the bicycle as a case study for their argument for the social construction of technology. Latour (1992) argues that seatbelts and the "Berliner key", which requires one to close and lock the door behind oneself in order to retrieve the key, are delegations of human function and interest to an artifact. Weber (1985) describes the policy process whereby the U.S. Airforce altered height requirements in order to accommodate female pilots, who previously had been thought to be unsuited to the task. And Friedman and Nissenbaum (1997) identified numerous cases of "bias" in computer systems.

Clearly, technology design is an appropriate subject for policy analysis. Particularly for artifacts like automobiles, nuclear reactors, and bridges. But what of a filing system? Does a schema for

categorizing things deserve scrutiny? In *Sorting Things Out: Classification And Its Consequences*, Bowker and Star (1999:33) argue they do: "Systems of classification (and of standardization) form a juncture of social organization, moral order, and layers of technical integration." Bowker and Star described how a nursing intervention system was altered to recognize that the time spent with patients was an important activity, rather than an inefficiency:

> Information, in Bateson's famous definition, is about differences that make a difference. Designers of classification schemes constantly have to decide what really does make a difference; along the way they develop an economy of knowledge that articulates clearance and erasure and ensures that all and only relevant features of the object (a disease, a body, a nursing intervention) being classified are remembered. (Bowker and Star 1999:281)

Or, as Reagle (1999) noted in *Eskimo Snow and Scottish Rain: Legal Considerations of Schema Design*,

> In Judeo-Christian theology the first power granted by its God to man was the power to name, "Out of the ground the Lord God formed every beast of the field and every bird of the air, and brought them to Adam to see what he would call them. And whatever Adam called each living creature, that was its name." (Gen2:19-20). Designing a schema that others will use is -- in some sense -- an exercise of the power to name.

The example of Pluto being deprecated from the category of planet is a recent example of how contentious categorization can be. Designing the categories by which we interact with each other and our systems is bound to privilege some point of view, while muting others. Yet, not every system is a tool of sinister hegemonic forces with social implications far outstripping its technical scope. Sometimes the technology is very simple, as is its interface to the social world. How then, might one come to understand bug and issue tracking systems?

2.2 Bug Tracking

A bug tracking system is simply an issue tracking system about software bugs. Subsequently, I will use the term "bug tracking" generically unless there is cause to make a distinction. The reason I opt for "bug" over the more generic "issue" is because bug tracking systems are prominent in public use and as objects of discussion, and in practice many bug tracking systems track more than software bugs: they might also include proposals for new features (i.e. a wishlist).

One of the most well known bug tracking systems is Bugzilla. It is an open source project used to track bugs of other open source projects, most notably the Mozilla Web browser, a descendant of the Netscape browser. Open source projects produce software that is available in source code form and amendable to modification by others. Typically, the work process is open as well, so one can follow the discourse of the community in their e-mail, chat, or bug tracking conversations. Bugzilla (Mozilla 2002) describes itself as follows:

> Bugzilla is a database for bugs. It lets people report bugs and assigns these bugs to the appropriate developers. Developers can use Bugzilla to keep a to-do list as well as to prioritize, schedule and track dependencies.... Enter the tasks you're planning to work on as enhancement requests and Bugzilla will help you track them and allow others to see what you plan to work on. If people can see your flight plan, they can avoid duplicating your work and can possibly help out or offer feedback.

The Bugzilla system is a tool for collaboration, if for no other reason than to help avoid duplicate work. Shukla and Redmiles (1996) provide a succinct summary of the bug tracking process as a collaborative learning process and identify the stakeholders, including end-users, designers, implementers, and management. Additionally, sometimes a "bug-czar" or "quality assurance" person facilitates the processing of a bug through its "life cycle." Finally, while anyone could theoretically fix a bug, there is often a small group of individuals responsible for portions of code. A fix, or "patch," often comes from the core group since they know the code the best, or have the authority to mediate access to that code in the community's software versioning system.

In Bugzilla, when bugs are first submitted they are categorized as UNCONFIRMED, "this means that a QA (Quality Assurance) person needs to look at it and confirm it exists before it gets turned into a NEW bug" (Bugzilla 2004). When a bug is fixed it is marked as RESOLVED and given a resolution specified in (Bugzilla 2004):

> FIXED: A fix for this bug has been checked into the tree and tested by the person marking it FIXED.
>
> INVALID: The problem described is not a bug, or not a bug in Mozilla.
>
> WONTFIX: The problem described is a bug which will never be fixed, or a problem report which is a "feature", not a bug.
>
> LATER and REMIND: These are both deprecated. Please do not use them.
>
> DUPLICATE: The problem is a duplicate of an existing bug. Marking a bug duplicate requires the bug number of the duplicating bug and will add a comment with the bug number into the description field of the bug it is a duplicate of.
>
> WORKSFORME: All attempts at reproducing this bug in the current build were futile. If more information appears later, please re-open the bug, for now, file it.
>
> MOVED: The bug was specific to a particular Mozilla-based distribution and didn't affect mozilla.org code. The bug was moved to the bug database of the distributor of the affected Mozilla derivative.

When a QA person has confirmed the processing of a bug, the bug is marked as VERIFIED. When the software is "shipped" (the corrected version is available to end users) it is marked CLOSED though it may be REOPENED at a later time. As is evidenced by the number of categories and the deprecation of LATER and REMIND resolutions, this typology and process of tracking the bugs has evolved according to the experiences of the users of the system. Most bug tracking systems work in a similar way though there will be differences in their typology and processes.

While I am not able to provide a historical treatment of how the specific Bugzilla categories and processes came to be as they are shown above, I will identify some of the tensions that have prompted the development of such categories more generally and how those tensions are the subject of specific debates today. But to do this, I first want to briefly consider the types of values that might be embedded in the design of a bug tracking system.

2.3 Discursive Values in a Public Sphere

Bug tracking tools mediate a conversation between the user and the developer; the developer is responsible for addressing the item raised by the user. These designations are *roles*, for any person might easily be both a user and developer of a piece of software. (In fact, developers

frequently file reports against their own code.) These conversations are civil for the most part; for, unlike other scenarios such as a zero-sum trade dispute, both parties have substantive interests in common. It is in the user's interest to not encounter bugs; this is also in the developers' interest with respect to his own satisfaction and as a fellow user.

However, there can be differences between the roles. There may be particular bugs or features that a user wants fixed that is not a priority to the developer – she has her own interests and there is only so much time in a day. Or, when pressed for time or feeling confused about who is responsible for a particular bug, a user might submit a less than complete bug report.

Jürgen Habermas has influenced understandings of civic discourse with the concept of the *public sphere*, "a domain of our social life in which such a thing as public opinion can be formed" (Habermas 1991:398). While this framework seems rather disproportionate to small-scale discussions about software bugs, such a theory can provide characteristics of (sometimes contentious) discourse that are relevant to the questions I'm asking. For example, in *Normative Criteria for the Public Sphere,* Ferree et al. (2002) describe four forms of discursive tradition:

- representative liberal: elite dominance, expertise, and proportionality; a free marketplace of ideas with transparency, detachment, civility; with an outcome focused on closure (p. 206)
- participatory liberal: popular inclusion; empowerment with a range of communicated styles; avoidance of a premature closure (p. 210)
- discursive: popular inclusion; empowerment with a focus on dialog, mutual respect, civility (though impassioned) and merit based decisions; closure is contingent on consensus (p. 215)
- constructionist: privileges the periphery and oppressed; with a communicative narrative of empowerment; avoidance of premature closure (p. 222).

In some ways, this typology is inappropriate for the sort of technical conversations that are the subject of this paper because the voluntary character of open source development permits a different sort of relationship between discussion and action. In civic discourse, public opinion relates to governmental action via one of the forms above. In free/open source discourse, developers can and do argue that they need only satisfy themselves, those who disagree can do it their own way as well. (If it is a complement to what another developer has already done, it can be added; if it is an alternative, it will vie for adoption as a competitive "fork"). Yet this is a value itself – one sympathetic to the voluntary nature of much of the development. In cases when the community does want to condense a collection of opinion into a single policy many of the variables above, such as elite dominance, expertise, and transparency, are relevant to the analysis. In any case, the elements of each form are relevant, even if, for example, it is difficult to identify a perfect example of the constructionist form in bug tracking discourse. In the next section I present some real world cases in which these values are reflected and discussed in the context of bug tracking systems.

3. Method

This analysis is based on participation in the Web development community. Of most relevance to this paper, I was a user and bug reporter of various open source Web browsers; specifically, I followed the development of KDE's Konqueror browser (and desktop) and Apple's Safari browser, which was built upon Konqueror's open source HTML rendering engine. Ethnographic and archival data for this paper spans, roughly, three years (2000-2003) during which there was

much discussion of bug tracking strategies and the implementation of new tools. Sources include discussions from bug tracking systems, developer mailing lists and blogs, and a KDE news portal and discussion site. I did not attempt to interview participants, but instead, simply acted as one, while also making notes of my experiences: "A culture is expressed (or constituted) only by the actions and words of its members and must be interpreted by, not given to, a field worker" (Van Maanen 1988). All cited discourse is public and can be easily accessed on the Web.

4. Findings: Values, Strategies, and Voting

4.1 Values of Software Development and Bug Tracking

The very openness and explicitness of these Web browser bug tracking systems demonstrates a valuation of the principal of transparency. However, one must be careful in inferring intention on the part of designers towards a particular value. Langdon Winner (1986:29) argues that technologies like nuclear power are "inherently" political as they depend on particular types of political relationships. While this is a valuable insight, I am concerned more generally with the "social" and would avoid the essentialist characterization of "inherent." Instead, in many information designs, some technical values might be "sympathetic" to particular social values. For example, Lawrence Lessig (1999) discusses the technical benefits of the end-to-end architecture of the Internet, as well as the civil consequences this architecture had in facilitating free expression. Some might then infer that the designers of the Internet or Web started their projects with emancipatory purpose. Perhaps, but it might also be that this was an unintended consequence, a serendipitous consequence, or something which was not considered at all. (Such emancipatory inferences about intention are often made with the benefit of hindsight.) This is what then leads Lessig to argue that if we wish to preserve the values of the original Internet (both the open architecture and freedom of expression) we can no longer rely solely upon this sympathetic relationship because both the technology and social norm can come under attack; one should persist in open technical designs, and support freedom enhancing laws and social norms.

A critical and difficult job in the open software world is to compile the source code into easily installable *packages* that are then available as a *distribution* to the end users. This job is difficult for a number of reasons. The first of which is in managing dependencies. A benefit of open source development is that applications can share modular software functionality; yet, the ways in which these applications depend upon each other across multiple versions can be complicated. For example, a windowing desktop might depend on version 1.0 of graphical library to render the icons, but the latest version of a popular puzzle game might require version 2.0 of that same library! These two applications cannot easily coexist. When such problems occur the user is most likely to vent their frustrations upon the package maintainers, which is further complicated in that they may be the inappropriate recipient of the bug: it might be a problem with the package, but it also might be a bug in the original the source code.

The difficulties of this job are apparent in the Debian KDE desktop packaging community. (KDE is a windowing environment; Debian is a Linux distribution of easy to install packages or "debs.") In response to challenges about how the dependencies of a package were being handled, the package maintainer, Ivan Moore (2000a), responded "I'm really getting tired of this… I had to cut down on the number of bug reports I was getting and verify that the packages worked or didn't work." Eventually, Moore declared that he would stop maintaining the packages; Erik Severinghaus (2000) posted Moore's announcement to a KDE community Web site and editorialized:

> This happened with freshmeat.net a while ago, it has happened to countless projects,

> and I'm *tired* of dumbasses flaming developers/packagers/webmasters/whatever who volunteer their time to OpenSource projects. Stop bitching and fix it.

The next day Moore (2000b) relented:

> just a note. I have gotten a ton of email from alot of people who are upset about this. So far none of it negative. I want to make it clear that the negative comments come from about 1% of the community...it's just that this 1% is always the percentage that is the loudest. This only because they are saying that what you are doing is bad or wrong or [insert negative comment here]... Anyways...because of all the nice comments I had decided to make the KDE 2.0 potato debs available...or rather continue to make them available.

Yet, a similar event caused Moore to finally resign in January of 2002. The following year, Chris Cheney, one of Ivan's successors, was challenged for his performance and (presumed) inexperience. Charles de Miramon (2003) responded to the complaint as follows:

> I resent your ageism intruding into this. Chris is an excellent maintainer. Just because he doesn't have the time to answer the same question repeatedly to people who would rather complain than either fix the problem, or accept that they've done far, far less for the Debian KDE community than Chris, doesn't make him a bad maintainer… If you're so fanatical about this, go do something. Make a website. Talk to debian-desktop. Create a metapackage, whatever. It's more productive than the email you just sent.

From these threads we can clearly identify the values from Ferree et al. of resource efficiency (minimizing expended time), expertise (the ability/merit of the maintainer and user), proportionality (the effect the 1% minority might have on morale), self-reliance/commitment (exhorting others to contribute), and mutual respect (providing positive feedback when needed).

4.2 Wizards and Strategies

In September 2002, the KDE bug system was switched over to a Bugzilla implementation with a KDE specific five-page bug reporting wizard. Prior to the use of Bugzilla and wizard, bugs were submitted via a single complex form. In an effort to encourage complete and unique bug reports, the wizard requires the completion of information such as a version and distribution, and presents the user with a set of existing bug reports that may be relevant. However, some frequent users considered the five-page wizard to be tedious. (The danger is that if a system is difficult to use, it can yield fewer legitimate reports.) Sebastian Laout (2004) submitted a bug report against the wizard itself: "Posting a bug in bugs.kde.org is a pain" and included a step-by-step analysis of the inefficiencies of the wizard process. However, presently, the bug's status is RESOLVED with a resolution of WONTFIX. Daniel Naber responded, "We *need* the wizard so that people stick at least to *some* rules. Otherwise we will drown in duplicates and reports that are even worse than now. If you have a better idea for the wizard, send patches." This is again demonstrative of the values of efficiency and self-reliance/commitment.

However, even within a perfectly efficient bug reporting system, the tension of differing priorities would remain. Dave Hyatt (2003), a lead developer of the Safari Web-browser for the Macintosh, noted an amusing strategy of bug submitters vying for developer attention:

> I love the tactics some people use when filing bugs. In particular the tactic of saying something inflammatory in order to goad the receiver of the bug into fixing it. You see this a lot in Bugzilla, and also in reported Safari bugs.

> Here are some of my favorite phrases (for your enjoyment). Let X = the browser of your choice. Let Y = any other browser.
>
> (1) The Promise - "The lack of this feature is the one thing that keeps me from switching to X."
>
> (2) "I can't work under these conditions. I'll be in my trailer." - "I can't believe you broke this! That's it! I'm going back to Y!"
>
> (3) Playing the EOMB Card - "How can this be broken? Every other modern browser gets this right."
>
> (4) Impatience - "Months have passed, and this bug still hasn't been fixed! What's the holdup?"
>
> (5) Overeagerness - "Still broken." (2 days later.) "Still broken." (2 days later.) "Feature still doesn't work. (2 days later.) "Broken in build from mm/dd/yy."
>
> The Safari team has actually started using the term EOMB as a way of referring to all other modern browsers. ;)

In order to give a voice to the user community, and limit minorities from using morale damaging strategies, some free/open software communities have implemented bug voting schemes.

4.3 Voting and "Democracy"

In a typical bug voting scheme, each registered reporter is allocated a fixed amount of points that they can spend on bugs, up to some ceiling per bug or application. The front page of the KDE bug reporting system includes reports such as weekly summary statistics, the most hated bugs, the most wanted features, the most frequently reported bugs, report counts by ownership, severity, and priority. (An additional feature of Bugzilla is that an UNCONFIRMED bug with a sufficient number of votes can be automatically elevated to NEW without the intervention of a quality assurance person.) This model is reminiscent of Ferree et al.'s "representative liberal" form wherein the media serves the purpose of ensuring the accountability of the representatives via transparency. Yet, different communities interpret the meaning of votes differently. Or, as Brey (1997) argues technical systems are subject to "different interpretations, not only of its functional and social-cultural properties but also of its technical content, that is, the way it works" (Brey 1997).

The Mozilla community quality advocate, Asa Dotzler (2002), has stated, "Votes aren't ignored but at the same time they're not the deciding factor in what gets fixed." He noted that votes are disproportionately spent on feature requests, disadvantaging critical bug reports; that those who file bug reports are a tiny fraction of all Mozilla users; and he argues bug reporters are probably not representative of the larger community. Furthermore, the voting scheme is simplistic (e.g., users can't vote against a feature).

Another common point of discussion is whether one should solicit others to vote on a particular bug. Aaron Seigo (2003) objected to this practice:

> if i may suggest, the best way to make voting on bugs.kde.org absolutely worthless is to recruit people wholesale to vote for various random bugs by posting them off-topic to places such as theDot. such campaigning distorts the statistical relevance inherent in the process. while you may achieve a surge in votes for your pet bug, you'll be doing a disservice to all the other bugs that have garnered votes "the hard way" even though

> those votes are probably much more relevant/important

Such a viewpoint represents a pluralistic view of a public sphere: each user should represent her own position, and the role of compromise and representation is seen as ultimately corrupting. Some participants note that these discussions begin to take on the character of "real-world" politics:

> "IMHO this is getting as annoying as the campaigning of political parties" (Loose 2003).
>
> The comparison to elections and advertising is truly astonishing given that campaign reform and an attempt to end undue influence returning to a "one person, one vote" ideal has been at the forefront of politics for years (Laffoon 2003).

An interesting issue that arises when one attempts to assess, for one's own satisfaction, which position on voting is "correct" is that there is no right or wrong; instead, what can be important for the cohesion of the community is the degree to which one of those interpretations is commonly held.

5. Conclusion

Bug tracking systems are, at first glance, seemingly boring and of little relevance on questions of community and discourse. On second glance, they might be seen as a media through which the community discusses and prioritizes issues important to it, but only in a narrowly technical way. In this paper I show that bug tracking systems mediate tensions between members of a software community. Adopting Ferree et al.'s "normative criteria for the public sphere" I identify within the KDE community the importance of the values of resource efficiency, expertise, proportionality, self-reliance/commitment, and mutual respect. When the KDE community became aware of the tensions between stakeholders and such values (e.g., users attempting to receive attention and developers responding "do it yourself") they deployed mechanisms such as bug voting. However, this prompted discussion on the appropriateness of campaigning and vote trading! From this, I conclude that this case exceeds the theoretical framework of "embedded," "embodied" (Grint and Woolgar 1995) or "inscribed" (Latour 1992) values. Instead, this case highlights the importance of ongoing interpretation (Pinch and Bijker 1992) in understanding the *meaning* of technology -- going beyond designers' intention.

References

Brey, P. 1997. “Philosophy of technology meets social constructivism,” *Techne: Journal for the Society for Philosophy and Technology:* 2(3-4).

Bork, J. 2003. “Anatomy of a bug,” Incessant Ramblings, http://headblender.com/joe/blog/archives/microsoft/001280.html

Bowker, G. C., and S.L Star. 1999. *Sorting things out: classification and its consequences*. Cambridge, MA: MIT Press. http://www.istl.org/00-winter/review2.html

Dotzler, A. 2002. “Mozilla 1.2.1 Coming Soon,”mozillaZine, http://www.mozillazine.org/talkback.html?article=2702

Ferree, M. M., W.A. Gamson, J. Gerhards, D. Rucht. 2002. “Normative criteria for the public sphere.” In *Shaping Abortion Discourse*. Cambridge: Cambridge University Press.

Friedman, B. and Nissenbaum, H. (1997). “Bias in computer systems,” In B. Friedman, ed., *Human Values and the Design of Computer Technology,* New York: Cambridge University Press, 21-40.

Grint, K., and S. Woolgar. 1995. "On some failures of nerve in constructivist and feminist analyses of technology," *Science, Technology, and Human Values,* 20: 286-310.

Hyatt, D. 2003. "Bug Guilt Trips," Surfin' Safari, http://weblogs.mozillazine.org/hyatt/archives/2003_11.html#004358

KDE News. 2002. "KDE Switches To Bugzilla,"http://dot.kde.org/1032319933/

Laffoon, E. 2003. "Re: OT: pls vote for this bug,"KDE News http://dot.kde.org/1058371499/1058390231/1058392962/1058425970/1058461094/

Latour, B. 1992. "Where are the missing masses? The sociology of a few mundane artifacts," In W. Bijker and J. Law, eds., *Shaping Technology/Building Society,* Cambridge, MA: MIT Press.

Lessig, Lawrence. 2000. *Code and Other Laws of Cyberspace,* New York: Basic Books.

Loose, C. 2003. "Re: OT: pls vote for this bug," http://dot.kde.org/1058371499/1058390231/1058392962/1058425970/1058429014/

Malone, T. 1998. "Commentary on uchman article and Winograd response." In B. Friedman, ed., *Human Values and Design of Computer Technology*, chapter 6, Oxford: Cambridge University Press.

Michael. 2004. "Announcing the KDE Quality Team Project," http://slashdot.org/article.pl?sid=04/03/02/1924204

Moore, I.E. 2000a. "Re: KDE & apt,"http://lists.debian.org/debian-devel/2000/debian-devel-200010/msg00445.html

Moore, I.E. 2000b. "Re: KDE Linux Packaging Project Taken Down," http://dot.kde.org/971680096/971755828/

Mozilla. 2002. "Bugzilla,"http://www.bugzilla.org/

Pinch, T., and W. Bijker. 1992. "The social construction of facts and artifacts: or how the sociology of science and the sociology of technology might benefit each other." In W. Bijker and J. Law, eds., *Shaping Technology/Building Society,* Cambridge, MA: MIT Press, 17-50.

Reagle, J. 1999. "Eskimo Snow and Scottish Rain: Legal Considerations of Schema Design. Reference: W3C Note 10-September-1999," http://www.w3.org/TR/1999/NOTE-md-policy-design-19990910.html

Seigo, A. 2003. "Re: OT: pls vote for this bug," http://dot.kde.org/1058371499/1058390231/1058392962/

Severinghaus, E. 2000. "Re: OT: pls vote for this bug," http://dot.kde.org/1058371499/1058390231/1058392962/

Shukla, S., and D. Redmiles. 1996. "Collaborative learning in a bug-tracking scenario," In *Conference on Computer Supported Cooperative Work* (CSCW 96), Association for Computing Machinery.

Suchman, L. 1998. "Do categories have politics? The language/action perspective reconsidered," In B. Friedman, ed., *Human Values and Design of Computer Technology*, chapter 4, Oxford: Cambridge University Press.

van Maanen, J. 1988. *Tales of the field: on writing ethnography,* Chicago: University Of Chicago Press.

Weber, R. 1985. *Manufacturing gender in the cockpit design,* Open University Press.

Winner, L. (1986). "Do artifacts have politics?" In *The Whale and the Reactor*, pages 18-39. The University of Chicago Press, Chicago.

Winograd, T. 1998. "Categories, disciplines, and social coordination," In B. Friedman, ed., *Human Values and Design of Computer Technology*, chapter 5, Oxford: Cambridge University Press.