

Politeness as a Social Software Requirement

Brian Whitworth, Massey University, Auckland, New Zealand

ABSTRACT

If politeness makes society a nicer place to be, by lubricating the interaction of its human parts, then the same is important for online society. As the Internet becomes more social, software can mediate social interactions, serve as a social agent or act as a personal assistant, but to succeed in these roles it must learn a new skill - politeness. This article proposes politeness as the distinguishing mark of a new generation of community software based on the benefits of social synergy rather than technical efficiency. Conversely, selfish software is currently a widespread problem as politeness is a software design "blind spot". An informational definition of politeness as the giving of choice suggests social software should be: 1. Respectful, 2. Transparent, 3. Helpful, and 4. Personal and 5. Responsive. For the Internet to realize its social as well as technical potential, software must be not only useful and usable but also polite. [Article copies are available for purchase from InfoSci-on-Demand.com]

Keywords: Agent; Computer Performance; It Evaluation Methods; Requirements Analysis; Socio-technical Design; System Errors; Technology Trends

INTRODUCTION

Computer Agents

Software, with its ability to make choices, seems to have crossed the threshold from inert machine to interaction participant, as the term human-computer interaction (HCI) implies. Computers today are no longer just tools that respond passively to directions, but agents, assistants and, in general, online participants in their own

right. Miller notes that if I accidentally hit my thumb with a hammer, I blame myself not the hammer, yet people may blame an equally mechanical computer for errors they initiate (Miller, 2004, p. 31). Computers are just as mechanical as cars, but while a car inertly reflects its driver's intentions, computers now ask questions, request information, suggest actions and give advice. Nor are computers mediating a social interaction, like email, simply passive, as the software, like a facilitator, affects the social

interaction possibilities (Lessig, 1999). As computers evolve, people increasingly find them active collaborators and participators, rather than passive appliances or media. In these new social roles, as agent, assistant or facilitator, software has a new requirement – to be polite.

If software can be social it should be designed accordingly. A company would not let a socially ignorant person represent it to important clients. Yet often today's software interrupts, overwrites, nags, changes, connects, downloads and installs in ways that annoy and offend users (Cooper, 1999). While such behaviour is not illegal it is certainly impolite.

Selfish Software

The contrast to polite software is “selfish software”. Like a selfish person who acts as if only he or she exists, so selfish software acts as if it were the only application on your computer. It typically runs itself at every opportunity, loading at start-up, and running continuously in the background. It feels free to interrupt you any time, to demand what it wants, or announce what it is doing, e.g. after installing new modem software it then loaded itself on every start-up, and regularly interrupted me to go online to check for updates to itself. It never found any, even after many days, so finally after yet another pointless “Searching for upgrades” message I decided to uninstall it. As in “The Apprentice” TV show, the reaction to assistants that don't do what you want is: “You're fired!”

If impolite software can drive users away, this implies a new type of software error – a social error. When a program gets into an infinite loop that “hangs” the computer the software has created an information processing error. When poor usability

means a user cannot operate a computer system, the software has created a human processing error. When software socially offends and drives away users however it is a social error. In the case of a usability error, users want to use the system but don't know how to. In contrast for the case of impolite software users understand it all too well and choose to avoid it. Modern socio-technical systems cannot afford social errors as without social participation they fail. In practical terms a web site that no-one visits is as much a failure as one that crashes. Whether a system fails because the computer can't run it, the user doesn't know how to run it, or the user doesn't want to run it doesn't matter because the end effect is the same - the application doesn't run.

For example the author's new 2006 computer came with McAfee Spamkiller which when activated then deliberately (by design) overwrote my Outlook Express mail server account name and password with its own values. After checking why I could no longer receive mail I found my mail server account details were wrong, so retyped in the correct values to fix the problem and got my mail. However next time the system rebooted, McAfee rewrote over my mail account details again. The McAfee help person explained that Spamkiller was protecting me by taking control, and routing all my email through itself. To get my mail I had to go into McAfee and tell it my specific email account details. That when I did this it didn't work is less the issue than why this well known software:

- a. Felt entitled to overwrite the email account details a user had typed in.
- b. Could not copy my account details, which it wrote over, to create its own account.

This same software also “took charge” whenever Outlook started, forcing me to wait as it did a slow foreground check for email spam. Yet in two weeks of use, it never found any spam at all! Again, not being hostage to this software, I concluded it was selfish software and uninstalled it.

Polite Computing

Politeness is distinct from usefulness and usability, where usefulness addresses system functionality and usability is how people use that functionality. The first focuses on what the computer does and the second on how users get the computer to do it. Polite computing in contrast is about social relations rather than computer power or cognitive ease. It addresses the requirement for a social entity to collaborate rather than compete with another party. Hence software can be easy to use yet rude, or polite but hard to use. While usability reduces training and documentation costs, politeness lets a software agent work with a competent user without frustration. Both usability and politeness however fall under the rubric of human-centred design.

Polite computing is about designing software to support politeness, not making people polite. People are socialized by physical society, but the widespread problem of software that is rude, inconsiderate or selfish is a general software design “blind spot” (Cooper, 1999) that cries out for rectification. Currently most software is socially “blind”, with the notable exception of socio-technical applications like Wikipedia, Facebook and E-Bay. This article aims to define, specify and illustrate an information based vision of polite computing for a new generation of social software.

THE VALUE OF POLITENESS

The Oxford English Dictionary (<http://dictionary.oed.com>) defines politeness as:

“... behaviour that is respectful or considerate to others”.

Taking politeness as consideration of the other in a social interaction, its predicted effect is a more pleasant social interaction. A general increase in politeness will be argued to make a society a nicer place to be, whether it is online or offline. Polite computing can contribute to computing by:

1. Increasing legitimate interactions.
2. Reducing anti-social attacks.
3. Increasing synergistic interactions.
4. Increasing software use.

There is nothing to stop programmers faking politeness, just as nothing stops people in the physical world from doing so, but when people behave politely, cognitive dissonance theory finds they also tend to feel more polite (Festinger, 1957). Likewise if programmers design for politeness, the overall effect will be positive, even though some may pretend.

Politeness and Legitimacy

Legitimate interactions, defined as those that are both fair and in the common good, have been proposed as the complex social source of civilized prosperity (Whitworth & deMoor, 2003), and a core requirement for any prosperous and enduring community (Fukuyama, 1992), while societies where unfair corruption and win-lose conflicts still reign are among the poorest in the world (Transparency-International, 2001). While

legitimate interactions are fair and in the public good, anti-social acts like theft or murder are unfair to victims and harm society overall. In crimes like theft or murder the victimized party usually has little or no choice, while in legitimate interactions like trade both parties can choose to trade or not. Hence one can argue that polite acts are more than fair, in that one party gives the other more choice than required. To do as the law requires is not politeness precisely because it is required, e.g. one does not thank a driver who stops at a red light, yet one thanks the driver who stops to let you into a line of traffic. While laws specify what citizens should do, politeness is about what they could do. If politeness involves offering more choices in an interaction than the law requires then it begins where fixed laws end. If criminal acts fall below the law, then polite acts rise above it, and polite, legitimate and anti-social acts can be ordered by the degree of choice offered to the other party or parties (Figure 1a). In this view politeness increases social “health”, just as criminality poisons it.

Politeness and Security

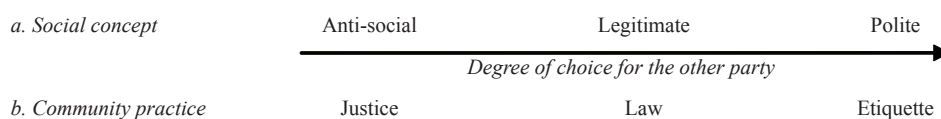
Even if polite computing has value, shouldn't it take a back seat to security issues? Yet upgrading security every time an attack exploits another loophole can be a never-ending cycle. An alternative is to develop strategies to reduce the motivation to attack society (Rose, Khoo, & Straub,

1999). Politeness can help one common source of attacks - resentment or anger against a system where the powerful are perceived to predate the weak (Power, 2000), and hacking is vengeance against a person, a company, or the capitalist society in general (Forester & Morrison, 1994). Politeness contradicts the view that since everyone takes what they can, so can I. That some people are polite, and give choice to others, may cause the neutral to become polite and those against society to become neutral. Politeness and security are then the two sides of the same coin of social health. By analogy, a gardener defends his or her crops from weeds, but does not wait for every weed to be killed before fertilizing. If politeness grows a better society, one should not wait to use it until every threat is purged. If security reduces anti-social acts, and politeness encourages social acts, they are complementary not mutually exclusive functions. Likewise diplomatic and military acts are viewed by most countries as complementary rather than mutually exclusive.

Politeness and Prosperity

Over thousands of years, as physical society became more “civilized”, this has created enormous prosperity, so for the first time in history some economies now produce more food than their people can eat (as their obesity epidemics testify). The bloody history of humanity seems

Figure 1. Community practice by social concept based on degree of choice given



to represent a social evolution from zero-sum (win-lose) interactions, such as war, to non-zero-sum (win-win) interactions, such as trade (Wright, 2001). Scientific research illustrates this social synergy, as for researchers to freely give their hard earned knowledge to all seems at first foolish, but when a critical mass do this, people gain more than they could have by working alone. Synergy means that when people in a community give to each other, they gain more than is possible by selfish activity, e.g. Open Source Software (OSS) products like Linux now compete with commercial products like Office. The mathematics of social synergy are that while individual gains increase linearly with group size, synergy gains increase geometrically, as they depend on the number of interactions not the number of group members. The Internet uses social synergy, as we each only “sow” a small part of it, but from it can “reap” the world’s knowledge.

Politeness and Usage

A study of reactions to a computerized Chinese word-guessing game found that when the software apologized after a wrong answer by saying “We are sorry that the clues were not helpful to you.” the game was rated more enjoyable than when the computer simply said “This is not correct” (Tzeng, 2004). In general, politeness improves the social interactions of a society which makes it a nicer place to be. The reader can judge for him or herself whether the World Wide Web is currently a nice place to be, or whether its “dark side”, which includes spam, spyware, viruses, hackers, pop-up ads, nagware, identity theft, solicitations, pornography, spoofers and worms (Power, 2000), means it could benefit from polite computing. If software were more polite,

people might be more willing to use it and less willing to abuse it.

AN INFORMATION DEFINITION OF POLITENESS

Social Software

To treat machines as people seems foolish, like talking to an empty car, but words seemingly addressed to cars on the road are actually to their drivers. While the cars are indeed machines, their drivers are people. Likewise while a computer is a machine, people “drive” the programs we interact with. Hence people show significantly more relational behaviours when the other party in computer mediated communication is clearly human than when it is not (Shectman & Horowitz, 2003), and studies find that people don’t treat computers as people outside the mediation context (Goldstein, Alσιο, & Werdenhoff, 2002)—just as people don’t usually talk to empty cars. Reacting to a software installation program as if to a person is not unreasonable if the program has a social source. Social questions like: “Do I trust you?” and “What is your attitude to me?” now apply. If computers have achieved the status of semi-intelligent agents, it is natural for people to treat them socially, and thus expect politeness.

A social agent is an interacting entity that represents another social entity in an interaction, either person or group, e.g. if an installation program represents a company (a social entity) the installation program is a social agent that interacts with the customer on behalf of the company. The interaction is social even if the agent is a computer, and an install creates a social contract even though the software is not social in itself. In the special case where a software agent works for the party it interacts with,

it is a software assistant, both working for and to the same user. In all such cases of human-computer interaction (HCI), social concepts like politeness apply.

Politeness as Niceness

Nass defines politeness as “being nice” to the other party (Nass, 2004), and argues that when another says “I think I’m a good teacher; what do you think?” polite people respond “You’re great”, even if they don’t think so. In this view, agreeing with another’s self praise is considered one of the “most fundamental rules of politeness” (Nass, 2004, p36). Yet while agreeableness may often accompany politeness, it does not define it if one can be both agreeably impolite and politely disagreeable. One can politely refuse, beg to differ, respectfully object and humbly criticize, i.e. disagree but still be polite. Conversely one can give charity to others yet be impolite, i.e. be kind but rude. Being polite is thus different from being kind, e.g. parents may be kind to a young child but not let it choose its own bedtime.

Politeness and Etiquette

To apply politeness to computer programming, it must be defined in information terms. If politeness is “considering others”, then since different societies “consider” differently, what is polite in one culture can be rude in another. Since there is no universal “polite behaviour”, there seems no basis to apply politeness to the logic of programming. Yet while different countries have different laws, the goal of fairness that underlies the law can be attributed to every society (Rawls, 2001). Likewise different cultures could have different “etiquettes” but a common goal of politeness. Figure 1b

distinguishes the specific implementations of systems of justice, legality and etiquette from the general social conceptual dimension of choice in the social interaction. In this view while societies “implement” different etiquettes, politeness is the common “design goal”, just as legitimacy is the “spirit” behind laws that vary in detail between societies. Indeed it is the social concept that allows a society to generate a new laws and new etiquettes for new situations.

If politeness can take different forms in different societies, to ask which implementation applies online is to ask the wrong question. This is like asking whether one country should adopt the laws of another. Laws must be “home grown” for each new case, so the right question is how to “reinvent” politeness in new online cases, whether for chat, wiki, email or other groupware. Just as different physical societies develop different local etiquettes and laws, so new online communities must develop their own ethics and practices, with software playing a critical support role. While different applications may need a different politeness implementations, it will be possible to develop general design “patterns” which specify politeness in abstract information terms (Alexander, 1964).

Politeness as Giving Choice

If the person considered knows what is “considerate” for them, politeness can be defined abstractly as the giving of choice to another in a social interaction. Doing this is then always considerate if the other knows what is good for them, though the latter assumption may not always be true, e.g. a young baby. In a conversation, where the locus of channel control passes back

and forth between parties, it is polite to give control to the other party (Whitworth, 2005), e.g. it is impolite to interrupt someone, as that removes their choice to speak, and polite to let them finish talking, as they then choose when to stop. This gives a definition of politeness as:

... any unrequired support for situating the locus of choice control of a social interaction with another party to it, given that control is desired, rightful and optional. (Whitworth, 2005, p355)

Unrequired means the choice given is more than required by the law, as a required choice is not politeness. Optional means the polite party has the ability to choose, as politeness must be voluntary. Desired by the receiver means giving choice is only polite if the other wants it, e.g. "After you" is not polite when facing a difficult task. Politeness means giving desired choices, not forcing the locus of control, with its burden of action, upon others. Finally, rightful means that consideration of someone acting illegally is not polite, e.g. to considerately hand a gun to a serial killer about to kill is not polite.

Impolite Computing

This definition suggests that impolite computing has a long history. Spam for example fills inboxes with messages users do not want (Whitworth & Whitworth, 2004), and is impolite because it takes choice away from email receivers. Pop-up windows are also impolite, as they "hijack" the user's cursor or point of focus, and take away the user choice of what they want to look at, hence many browsers now prevent pop-ups. Impolite computer programs:

1. Use your computer's services. Software can use your hard drive to store information cookies, or your long distance phone service to download without asking.
2. Change your computer settings. Software can change browser home page, email preferences or file associations.
3. Spy on what you do online. Spyware, stealthware or software back doors can gather information from your computer without your knowledge, or record your mouse clicks as you surf the web and, even worse, give your private information to others.

For example Microsoft's Windows XP Media Player, was reported to quietly record the DVDs it played and use the user's computer's connection to "phone home", i.e. send data back to Microsoft (Editor, 2002). Such problems differ from security threats, where hackers or viruses break in to damage information. This problem concerns those we invite into our information home, not those who break in, e.g. "software bundling", where users choose to install one product but are forced to get many:

When we downloaded the beta version of Triton [AOL's latest instant messenger software], we also got AOL Explorer – an Internet Explorer shell that opens full screen, to AOL's AIM Today home page when you launch the IM client – as well as Plaxo Helper, an application that ties in with the Plaxo social-networking service. Triton also installed two programs that ran silently in the background even after we quit AIM and AOL Explorer. (Larkin, 2005).

Likewise Yahoo's "typical" installation of their IM also downloads their Search Toolbar, anti-spyware and anti-pop-up software, desktop and system tray shortcuts, as well as Yahoo Extras, which inserts Yahoo links on your browser. It also alters the users' home page and auto-search functions to point to Yahoo by default. Even Yahoo employee, Jeremy Zawodny dislikes this:

I don't know which company started using this tactic, but it is becoming the standard procedure for lots of software out there. And it sucks. Leave my settings, preferences and desktop alone. (<http://jeremy.zawodny.com/blog/archives/005121.html>)

A similar scheme is to use security updates to install new products, e.g. "Microsoft used the January 2007 security update to induce users to try Internet Explorer 7.0 whether they wanted to or not. But after discovering they had been involuntarily upgraded to the new browser, some next found that application incompatibility effectively cut them off from the Internet." (Pallatto, 2007)

Security cannot defend against people one invites in, especially if the offender is the security system itself. However in a connected society social influence can be very powerful. In physical society the withering looks given to the impolite are not toothless, as what others think of you affects how they behave towards you. In traditional societies banishment was often considered worse than a death sentence. Likewise an online company with a reputation for riding roughshod over user rights may find this is not good for business.

SPECIFYING SOFTWARE POLITENESS

Based on previous work (Whitworth, 2005) it can be suggested that polite software is:

1. Respectful. Polite software respects user rights, does not pre-empt user choices, and does not act on or copy information without the permission of its owner.
2. Transparent. Polite software does not sneak or change things in secret, but openly declares what it does, who it represents, and how they can be contacted.
3. Helpful. Polite software helps users make informed choices, giving information that is useful to the other party.
4. Personal. Polite software remembers its past interactions with a user, and carries forward past choices to future interactions.
5. Responsive. Polite software responds appropriately to user actions, reflecting user directions rather than pursuing its own directions.

Each of these points is now considered in more detail.

Respectful

Respect includes not taking another's rightful choices. If two parties jointly share a resource, one party's choices can deny the other's, e.g. if I delete a shared file you can no longer print it. Polite software should not preempt rightful user information choices regarding common resources like the desktop, registry, hard drive, task bar,

file associations, quick launch and other user configurable settings. Pre-emptive acts, like changing a browser home page without asking, act unilaterally on a mutual resource and so are impolite.

Information choice cases are rarely simple, e.g. a purchaser can use that software but not edit, copy or distribute it. Such rights can be specified as privileges, in terms of specified information actors, methods, objects and contexts (Table 1) (Whitworth, 2006). To apply politeness in such cases requires a legitimacy baseline, e.g. a software provider has no right to unilaterally upgrade a computer the user owns (though the Microsoft Windows Vista End User License Agreement (EULA) seems to imply this). Likewise users have no right to unilaterally upgrade, as this edits the product source code. In such cases politeness applies, e.g. the software suggests an update and the user agrees, or the user requests an update and the software agrees (for the provider). Similarly while a company that creates a browser owns it, the same logic means users own data they create with the browser, e.g. a cookie. Hence software cookies require user permission,

and users should be able to view, edit or delete “their” cookies.

Transparent

Part of a polite greeting in most cultures is to introduce oneself and state one’s business. Holding out an open hand, to shake hands, shows that the hand has no weapon, and that nothing is hidden. Conversely, to act secretly behind another’s back, to sneak, or to hide ones actions, for any reason, is impolite. Secrecy in an interaction is impolite because the other has no choice regarding things they don’t know about. Hiding your identity reduces my choices, as hidden parties are untouchable and unaccountable for their actions. When polite people interact, they declare who they are and what they are doing.

If polite people do this, polite software should do the same. Users should see who is doing what on their computer. However when Windows Task Manager shows cryptic process like CTSysVol.exe, attributed to the user, it could be system critical process or one left over from a long uninstalled product. Lack of transparency is why after 2-3 years Windows becomes “old”. With

Table 1. Socio-technical actors, objects and methods

Actors	Objects	Methods
<i>People</i>	<i>Persona</i> (represent people)	<i>Create/Delete/Undelete</i>
<i>Groups</i>	<i>Containers</i> (contain objects)	<i>Edit/Revert</i>
<i>Agents</i>	<i>Items</i> (convey meaning)	<i>Archive/Unarchive</i>
	- <i>Comments</i> (dependent meaning)	<i>View/Hide</i>
	- <i>Mail</i> (transmit meaning)	<i>Move/Undo</i>
	- <i>Votes</i> (choice meaning)	<i>Display/Reject</i>
		<i>Join/Resign</i>
		<i>Include/Exclude</i>

every installation, selfish software puts itself everywhere, so the taskbar fills with icons, the desktop with images, the disk with files and the registry with records. Many applications consider themselves so important they need to load at start-up and run continuously just in case you need them. When many applications do this it slows down the computer considerably. Taskbar icon growth is just the tip of the iceberg of what is happening to the entire computer, as many start-ups don't show on the taskbar. Because selfish programs put files wherever they like, uninstalled applications are not removed cleanly, and over time Windows accretes an ever increasing "residue" of files and registry records left-over from previous installs, until eventually only reinstalling the entire operating system recovers system performance.

The problem is that the operating system keeps no record to make who does what transparent to the user. An operating system Source Registry could link all online technical processes to their social sources, giving contact and other details. "Source" could be a property of every desktop icon, context menu item, taskbar icon, hard drive file or any other resource. A user could delete all resources allocated by a given source without concern that they were system critical. Windows messages could also state their source, so users know who a message is from. Application transparency would let users decide what to keep and what to drop.

Helpful

A third politeness property is to help the user by offering understandable choices, as a user cannot properly choose from options they do not understand. Offering

options that confuse is inconsiderate and impolite, e.g. a course text web site offers the choices:

- OneKey Course Compass
- Content Tour
- Companion Website
- Help Downloading
- Instructor Resource Centre

It is unclear how the "Course Compass" differs from the "Companion Website", and why both seem to exclude "Instructor Resources" and "Help Downloading". Clicking on these choices, as is typical for such sites, leads only to further confusing menu choices. The impolite assumption is that users enjoy clicking links to see where they go. Yet information overload is a serious problem for web users, who have no time for hyperlink merry-go-rounds.

Yet to not offer choices at all on the grounds that users are too stupid to understand them is also impolite. Installing software can be complex, but so is installing satellite TV technology, and those who install the latter do not just come in and take over. They know that socially the user is in charge, and they expect to hear their choices presented to them in an understandable way or they may decide not to install. Complex installations are simplified by choice dependency analysis, of how choices are linked, as Linux's installer does. Letting a user choose to install an application they want minus a critical system component is not a choice but a trap. Application-critical components are part of the higher choice to install or not, e.g. a user's permission to install may imply access to hard drive, registry and start menu, but not to desktop, system tray, favourites or file associations.

Personal

It is not enough to give choices now but forget them later. If previous responses are forgotten, the user must redo them, which is inconsiderate. Hence software that actually listens and remembers past user choices is a wonderful thing. Polite people remember previous encounters, yet each time I open Explorer it fills its preferred directory with files I don't want to see, then returns the cursor to me to select the directory I want to look at, which is never the one displayed. Each time, Explorer acts as if it were the first time I had used it, yet I am the only person it has ever known. Why can it not remember where I was last time, and return me there? The answer is simply that it is impolite by design.

Such "amnesia" is a trademark of impolite software. Any document processing software could automatically open the user's last document, and put the cursor where they left off, or at least give that option (Raskin, 2000 p31). The user logic is simple: If I close the file I am finished, but if I just leave then put me back where I was last time. It is amazing that most software cannot even remember its last user interaction. Even within an application, like Outlook's email, if one moves from inbox to outbox and back, it "forgets" the original inbox message, and one must scroll back to it.

Responsive

Current "intelligent" software tries to predict user wants but cannot itself take correction, e.g. Word's auto-correct function changes $i = 1$ to $I = 1$, but if you change it back the software ignores your act. This is software that is clever enough to give corrections but not clever enough to take

correction itself. However responsive means responding to the user's direction not ignoring it.

It is not responsive to interrupt inappropriately, as this disturbs the user's train of thought. For complex work like programming, even short interruptions cause a mental "core dump", as the user drops one thing to attend to another. The interruption effect is then not just the interruption time, but also the user recovery time (Jenkins, 2006), e.g. if a user takes three minutes to refocus after an interruption, a 1 second interruption every three minutes can reduce productivity to zero. Mr. Clippy, Office '97's paper clip assistant, had this problem, since as one user noted: "It wouldn't go away when you wanted it to. It interrupted rudely and broke your train of thought." (Pratley, 2004). Searching the Internet for "Mr. Clippy" gives comments like "Die, Clippy, Die!" (Gauze, 2003), yet its Microsoft designer still wonders: "If you think the Assistant idea was bad, why exactly?" (Pratley, 2004). To answer simply, Mr Clippy was impolite, and in XP is replaced by polite smart tags. In contrast tag clouds and reputation systems illustrate software that reflects rather than directs online users.

Selfish software, like a spoilt child, repeatedly interrupts unnecessarily, e.g. Windows Update advises me when it starts, as it progresses, and when it finishes its update. Its modal window interrupts what I am doing, seizes the cursor and loses my current typing. Since each time Update only needs me to press OK, this is like being repeatedly interrupted to pat a self-absorbed kiddie on the head. The lesson of Mr. Clippy, that software serves the user not the other way around, seems still unlearned at Microsoft.

It is hard for selfish software to keep appropriately quiet, e.g. Word can generate a table of contents from a document's headings. However if one sends just the first chapter of a book to someone, with the book's table of contents (to show its scope), every table of contents heading line without a page number loudly declares: "ERROR! BOOKMARK NOT DEFINED", which of course completely spoils the sample document impression (Figure 2). Even worse, this effect is not apparent until the document is received. Why could the software not just quietly put a blank instead of a page number? Why must it announce its needs so rudely? What counts is not what the software needs but what the user needs, and in this case the user needs the software to be quite.

APPLICATIONS AND EXAMPLES

Blameware

A fascinating psychological study could compare computer messages when things are going well to when they are not. While computers seem delighted to take charge when things go well, when they go wrong software seems to universally agree that you have an error rather than we have an error. Brusque and often incomprehensible error messages like the "HTTP 404 – File not Found" imply that you need to fix the problem you have clearly created. Even though software itself often causes errors, most software feels no obligation to even

Figure 2. A book table of contents as emailed to a colleague (Word)

HANDBOOK OF RESEARCH ON SOCIO-TECHNICAL DESIGN AND SOCIAL NETWORKING SYSTEMS	1
Foreword	Error! Bookmark not defined.
Preface	Error! Bookmark not defined.
Acknowledgements	Error! Bookmark not defined.
Section 1: General socio-technical theory	Error! Bookmark not defined.
Prologue	Error! Bookmark not defined.
Chapter 1. The social requirements of technical systems	Error! Bookmark not defined.
Chapter 2. The Social Study of Computer Science	Error! Bookmark not defined.
Chapter 3. Virtual Collaboration and Community	Error! Bookmark not defined.
Chapter 4. The Social Derivation of Technical Systems	Error! Bookmark not defined.
Chapter 5. Socio-technical theory and work systems in the information age	Error! Bookmark not defined.
Chapter 6. An engagement strategy for community network research and design	Error! Bookmark not defined.
Chapter 7. On the Alignment of Organizational and Software Structure	Error! Bookmark not defined.
Section 2: Socio-technical perspectives	Error! Bookmark not defined.
Prologue	Error! Bookmark not defined.
Chapter 8. Privacy and the Identity Gap in Socio-Technical Systems	Error! Bookmark not defined.
Chapter 9. Privacy regulation in the Metaverse	Error! Bookmark not defined.
Chapter 10. Leadership of Integrated Teams in Virtual Environments	Error! Bookmark not defined.
Chapter 11. Recontextualising Technology in Appropriation Processes	Error! Bookmark not defined.
Chapter 12. Explaining Participation in Online Communities	Error! Bookmark not defined.
Chapter 13. Cyber Security and Anti-Social Networking	Error! Bookmark not defined.
Chapter 14. Emerging Cybercrime Variants in the Socio Technical Space	Error! Bookmark not defined.
Chapter 15. Developing innovative practice in Service Industries	Error! Bookmark not defined.
Section 3: Socio-technical analysis	Error! Bookmark not defined.
Prologue	Error! Bookmark not defined.
Chapter 16. Using communication norms in socio-technical systems	Error! Bookmark not defined.
Chapter 17. Socio-instrumental pragmatism in action	Error! Bookmark not defined.
Chapter 18. A Framework for Using Analytics to Make Decisions	Error! Bookmark not defined.
Chapter 19. The Challenges of Co-design and the Case of e-ME	Error! Bookmark not defined.
Chapter 20. Formal Analysis of Workflows in Software Development	Error! Bookmark not defined.
Chapter 21. The Role of Expectations in Information Systems Development	Error! Bookmark not defined.
Chapter 22. Building a Path For Future Communities	Error! Bookmark not defined.

give the information it has in a useful form, let alone suggest solution options for your error. In contrast polite software would see all interactions as involving “we”. Indeed studies of users in human-computer tutorials show that users respond better to “Lets click the Enter button” than to “Click the Enter button” (Mayer, Johnson, Shaw, & Sandhu, 2006).

Pryware

Pryware is software that asks for more information than it needs for any reasonable purpose, e.g. an online purchase needs an address to post to but an online registration to a special interest web site does not. Figure 3 shows a special interest site which wants people to register for no obvious gain and asks for not only name

and email but also work phone number, job description, institution, work address and skype telephone, and some sites are even more intrusive. When such unrequired fields are mandatory rather than optional people choose not to register, as while willing to give out name and email they much less willingly divulge home phone, cell phone or home address (Foreman & Whitworth, 2005), or they may record an address like “123 Mystreet, Hometown, zip code 246”. Polite software should not ask for unneeded information as a refusal can offend.

Nagware

If a choice repeats, to ask the same question over and over, for the same reply, is to pester or nag, like the “Are we there yet?” of children on a car trip. This forces

Figure 3. *Pryware*

The screenshot displays a web browser window with a 'USER ACCOUNT' registration form. The browser's address bar shows 'Element/Frequen...' and the page title is 'User account I...'. The form is structured as follows:

- Account information:**
 - Username:** A text input field with a small asterisk indicating it is required. Below it, a note states: "Spaces are allowed, punctuation is not allowed except for full stops, hyphens, and underscores."
 - E-mail address:** A text input field with a small asterisk. Below it, a note states: "A valid e-mail address. All e-mails from the system will be sent to this address. The e-mail address is not made public and will only be used if you wish to receive a new password or wish to receive certain news or notifications by e-mail."
- User details:**
 - Title:** A dropdown menu with radio button options: None, Assoc Prof, Dr, Miss, Mr, Mrs, Ms, and Prof. Below the options, it says "Choose your title".
 - First name:** A text input field with a small asterisk. Below it, a prompt says "Enter your first name".
 - Last name:** A text input field with a small asterisk. Below it, a prompt says "Enter your last name".
 - Work phone number:** A text input field with a small asterisk. Below it, a note states: "Enter your work phone number. Allowed characters are '0'-'9'".
 - Job title:** A text input field with a small asterisk. Below it, a prompt says "Enter your job title".

the other party to again and again give the same choice reply. Many users have still not updated to Windows Vista because of its reputation as nagware which asks too many questions. Polite people don't ask the same question over and over, yet software with no interaction memory does this, e.g. when reviewing email offline in Windows XP, actions like using Explorer trigger a "Do you want to connect?" request every few minutes. No matter how often one says "No!" it keeps asking, because the software cannot remember its own past. Yet software has already solved this problem, e.g. uploading a batch of files creates a series of "Overwrite Y/N?" questions to which would force the user to continuously reply "Yes", but there is a "Yes to All" meta-choice that remembers for the choice set. Such choices about choices (meta-choices) are polite. A general meta-choice console (GMCC) would give users a common place to see or set all meta-choices (Whitworth, 2005).

Strikeware

Strikeware is software that acts pre-emptively without asking on user resources. An example is a zip extract product that without asking put all the files it extracted as icons on the desktop! Such software tends to be used only once. Installation programs are notorious for pre-emptive acts, e.g. the Real-One Player adds desktop icons and browser links, installs itself in the system tray, and can commandeer all video and sound file associations. Customers resent such invasions, which while not illegal are impolite. An installation program changing your PC settings is like furniture deliverers rearranging your house because they happen to be in it. Software upgrades continue the tradition, e.g. Internet Explorer upgrades

that make MSN your browser home page without asking. Polite software does not do this.

Amnesic-Ware

It is astounding that major software manufacturers like Microsoft gather endless data on users, but seem oblivious to data on how their software interacts with the user. Like Peter Sellers in the film "Being There", such software "likes to watch" but struggles itself to relate to others. Someone should explain to these programmers that spying on users is not a user relationship, e.g. Mr Clippy watched your document actions but could not see his interactions with you, and so was oblivious to the rejection and scorn he evoked. Much software today is in the category of being less aware of their users than the average airport toilet. Hopefully tomorrow's software will make remembering user interactions its business, as its primary role is to work for people, not for itself.

Being amnesic makes the simplest of interactions difficult, e.g. Figure 4 shows what users must do to get a flash drive back from Windows XP after it has been put into a USB slot. Here is how this human-computer interaction would appear if one gave something to a human helper then asked for it back:

User: *Hi Bill, please sign this book (puts book in left hand)*

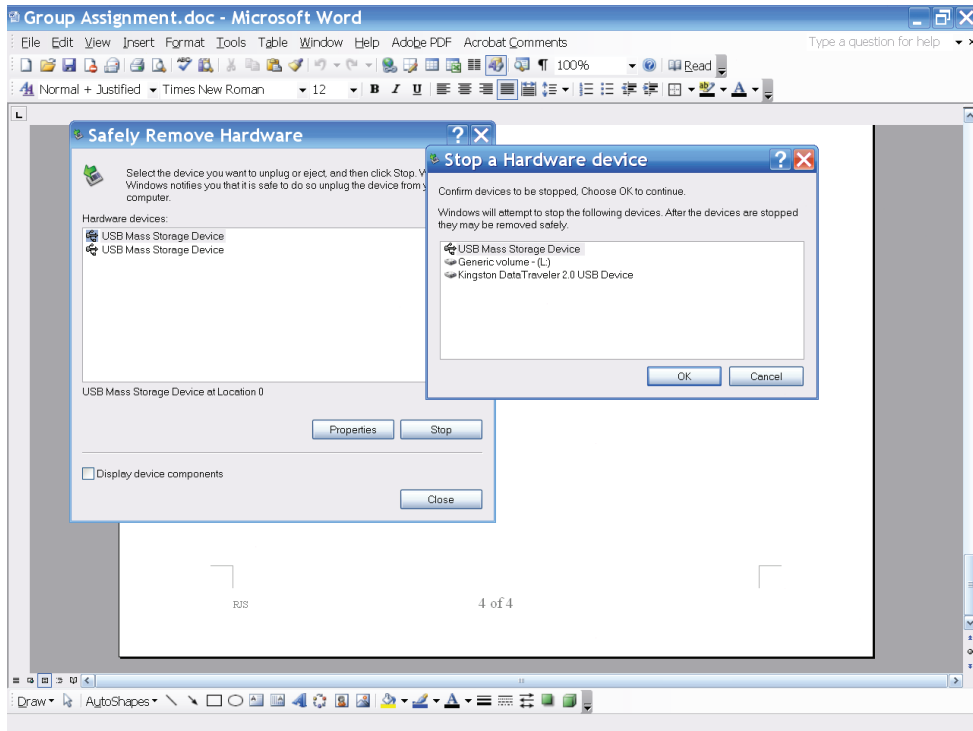
Bill: *Bill has found a book in his hand!*

User: *Give me the book back.*

Bill: *Do you want to safely remove the book in my left hand or my right hand?*

User: *I mean the book in your left hand Bill.*

Figure 4. Window's eject USB interface



Bill: *Confirm the book by pointing to it, or the hand, or to both, and say “OK”, and I will try to give it to you.*

User: *(points) OK*

Bill: *(left hand tries to give book but the right hand is holding it) I can't give it to you right now, try again later*

User: *Give me the book back!*

Bill: *Do you want to safely remove the book in my left hand or my right hand?*

A human helper who acted like this would be considered to have an attitude problem. If Windows can discover a USB is busy after the user selects it, why can't it check this before the user's selection? Is its time more important than that of the user? Most younger users of course just check

the USB drive light and if it is not flashing pull it out, avoiding all the above.

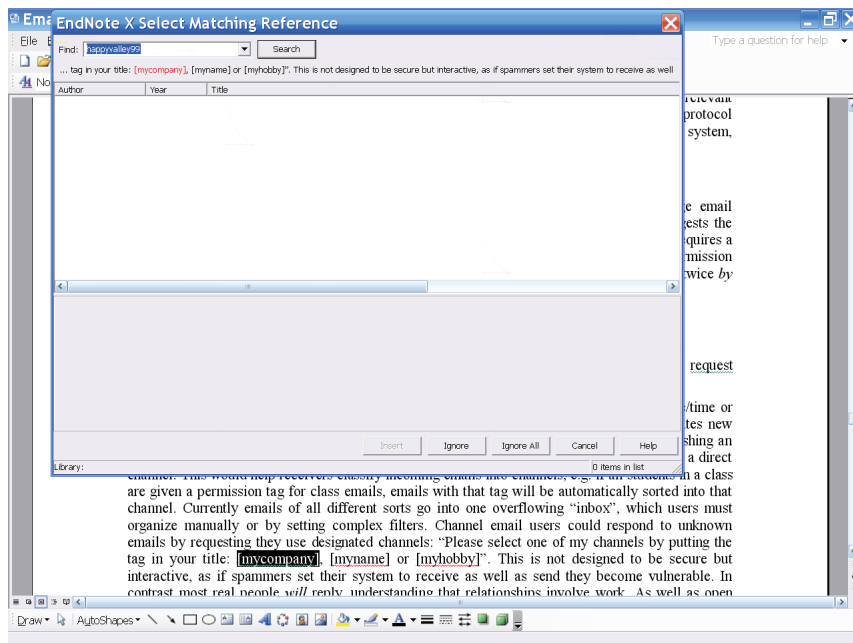
In a similar vein, every day people all over the world listen to slowly spoken computer telephone messaging reports like: “There are five new messages. The first message received at 12.15pm on Wednesday the 14th of November is “<click>” To save this message press 1, to forward it press 3, to reply to it press 5, ... to delete it press 76. The second message received at” Note that “76” as the code to press to delete a message (rather than say “1”) is the actual code the computer message system at my work uses. Again, imagine a human secretary who reported every detail before revealing the caller just hung up. These are not isolated cases, as inconsiderate software is a widespread problem

The Wizard's Apprentice

The problem presented here, of computers taking over what they don't really understand, is embodied in the folk story of the wizard who left his apprentice in charge but warned him not to try to use the magic. However as soon as he left, thinking he knew what he was doing, the apprentice started to cast spells which soon got out of hand and only the wizard's return prevented total disaster. Likewise as software takes over more it gets it wrong more. For example, Endnote software manages citations in documents like this one by embedding links to a reference database. Endnote Version X runs itself whenever you open the document, and if it finds a problem commands the focus from whatever you are doing to let you know right away (Figure 5). It uses square brackets for its links so assumes any

square brackets in your text are for it, just as selfish children assume any words spoken are to them. After telling it for perhaps the hundredth time to ignore brackets like these [], it then closes, dropping the cursor wherever it was and leaving you to find your own way back to whatever you were doing before it interrupted. There is no way to turn this Endnote activation off. Opening the document on another computer, such as at work, means Endnote can't find the right database. It handles this by clearing all the reference generating embedded links in the document, which the user then must re-enter manually. The only way I could stop this was to uninstall Endnote from my work machine. Conversely when you want to clear the links, as when trying to convert a Word/Endnote document to a publishing form, there is no way to clear the Endnote codes except manually. Selfish software,

Figure 5. Endnote X takes charge



like Mr Clippy, comes with everything but an off button.

There are many reasons why people should control computers, not the reverse. Firstly, while computers manage vast amounts of data with ease, they handle context changes poorly (Whitworth, 2008), so so-called “smart” computing invariably needs a human “minder”. Secondly, computers are not accountable for what they do, as they have no “self” to bear any loss. If society makes people accountable for what computers do, as it does, people need control over computer choices. Thirdly, the resistance of people to computer domination is predictable. Software designers should not underestimate the importance of user choice. In human history freedom and choice are the stuff of revolutions, and a grass-roots Internet movement against software arrogance is not inconceivable.

The future of computers lies not in becoming so clever or powerful that people are obsolete, nor in being passive human tools, but in contributing to a human-computer combination that performs better than either people or computers alone. The runaway IT successes of the last decade (cell-phones, Internet, e-mail, chat, bulletin boards etc) all support people rather than supplant them. As computers develop this co-participant role, politeness will be a critical success factor.

Today many users feel at war with their software: removing things they didn't want added, resetting changes they didn't want changed, closing windows they didn't want opened, and blocking e-mails they didn't want to receive, etc. User weapons in this unnecessary war, like third party blockers, cleaners, filters and tweekers, whose main aim is to put users back in charge of their computer estate, are the most frequent accesses at Internet download sites. If software

declares war on user choice it will not win, and if the Internet becomes a battlefield with choice the victim then no-one will want to go there. The solution is to give choices not take them, i.e. polite computing.

The Social Computing Revolution

Many successful online traders find politeness profitable. EBay's customer reputation feedback gives users optional access to valued information relevant to their purchase choice, which by the previous definition is polite. Amazon gives customers information on the books similar buyers buy, not by pop-up ads but as a view option below. Rather than a demand to buy, it is a polite reminder of same-time purchases that could save customer postage. Politeness is not about forcing users to buy, which is anti-social, but about improving the seller-customer relationship which leads to sales by giving customers choice, which is social. Polite companies win business because customers given choices come back. Perhaps one reason the Google search engine swept all before it was that its simple white interface, without annoying flashing or pop-up ads, made it pleasant to interact with. Google ads sit quietly at screen right, as options not demands. Yet while many online companies know that politeness pays, for others the lesson is still being learned, and for still others, hit-and-run rudeness remains an online way of life.

Wikipedia is just one example of the new generation of polite software. It does not act pre-emptively but responds to its users, it is transparent in what it does and supports transparency in interactions, it makes user acts like editing easy rather than throwing up obstacles and conditions, it remembers what each person does person-

ally, and it responds to user direction rather than trying to foist preconceived “good knowledge” conditions upon participants. Social computing features like post-checks (allowing an act then checking it later), versioning and rollback, tag clouds, optional registration, reviewer reputations, view filters and social networks illustrate how polite computing gives choices to people and responds to the choices of the people. In this movement from software autocracy to software democracy, the signature characteristic is a new politeness.

A Research Question

The users of modern software increasingly choose whether to use it or not, e.g. President Bush’s 2001 decision not to use e-mail because he did not trust it. That software can no longer hold users hostage to its power and users can now choose their preferred software gives a simple prediction: *Polite software will be used more and deleted or disabled less than impolite software.* Politeness here refers not just to the words used but also to the actions taken. Experimental studies could compare polite and impolite treatment conditions, correlational studies could compare application politeness with market success, longitudinal studies could discover if applications are becoming more polite over time, and ethnographic studies could explore how users perceive polite and impolite software. The scope of online politeness also bears investigation, as inexperienced users may tolerate impolite agents like Mr Clippy more than experienced users, and for interactions mandated by law politeness may not apply. Individual differences like gender, age and culture may mediate user reactions to impolite software. If future software suc-

cess depends on people volunteering to participate, then voluntary politeness may be the key to that success.

CONCLUSION

Polite computing suggests that giving choice is the new dimension of social computing. If so, software designers need to change their attitude to users. The first step is to stop seeing users as little children unable to exercise choice. While inexperienced users may let software take charge, experienced users want to make their own choices. The view that “software knows best” is hard to justify for today’s computer-literate users. If once users were child-like then today they have grown up.

The next step is for software to stop trying to go it alone. Too clever software assistants acting beyond their ability are already making core applications like Word like a magic world, where moved figures and titles suddenly jump about or even disappear entirely, resized column widths in tables reset themselves and text blocks change to entirely new formats like numbered after a delete. Increasingly only Ctrl-Z (Undo) saves the day, as it lets the user undo the too clever software’s errors. Software that hides its operations from the user then acts beyond its ability has misunderstood its role to be to lead, when really its role is to assist. Rather than using hidden logic to predict what users want before they know it themselves, why not follow the user direction? For example, I repeatedly change Word’s numbered paragraph default indents to my preferences, but it never remembers them. How hard is it to copy what the boss does? Even worse, it knows better, e.g. if I ungroup and regroup

a figure it takes the opportunity to reset my text wrap-around options to its inappropriate defaults, so the picture now overlaps the text again. Software should leverage user knowledge, with tool like the format paintbrush that can copy users layouts, not try to do it all itself.

Polite software does not act unilaterally, it transparently shows itself, it does not unnecessarily interrupt, it offers understandable choices, it remembers past interactions and it responds to user direction. Conversely, impolite software acts without asking, does things secretly, interrupts unnecessarily, offers confusing choices, has no recall of its past interactions and repeatedly goes its own way despite user corrections. It is not hard to imagine which type of software users prefer. A future is envisioned where politeness is a critical social requirement for software success. Polite computing could be taught in system design classes, along with other requirements. A "politeness seal" could credit applications that give rather than take user choice. For the Internet to realize its social as well as technical potential, software must become polite as well as useful and usable.

ACKNOWLEDGMENT

Many thanks to Guy Kloss, Massey University, for useful comments and insights.

REFERENCES

Alexander, C. (1964). *Notes on the Synthesis of Form*. Cambridge, Ma: Harvard University Press.

Cooper, A. (1999). *The Inmates are Running the Asylum- Why High Tech Products Drive us Crazy and How to Restore the Sanity*. USA.

Editor. (2002, Sunday, Feb 24, section 4). *Technology threats to privacy*. New York Times, p. 12.

Festinger, L. (1957). *A Theory of Cognitive Dissonance*: Stanford University Press.

Foreman, B., & Whitworth, B. (2005). *Information Disclosure and the Online Customer Relationship*. Paper presented at the Quality, Values and Choice Workshop, Computer Human Interaction (CHI).

Forester, T., & Morrison, P. (1994). *Computer Ethics* London: MIT Press.

Fukuyama, F. (1992). *The End of History and the Last Man*. New York: Avon Books Inc.

Gauze, C. F. (2003). *I See You're Writing an Article...* March. Retrieved 2006, Carl F. Gauze, from <http://www.ink19.com/issues/march2003/webReviews/iSeeYoureWritingAn.html>

Goldstein, M., Alsio, G., & Werdenhoff, J. (2002). *The media equation does not always apply: People are not polite to small computers*. *Personal and Ubiquitous Computing*(6), 87-96.

Jenkins, S. (2006). *Concerning Interruptions*. *Computer*, November, 114-116.

Larkin, E. (2005). *PC World*, December, 28.

Lessig, L. (1999). *Code and other laws of cyberspace*. New York: Basic Books.

Mayer, R. E., Johnson, W. L., Shaw, E., & Sandhu, S. (2006). *Constructing computer-based tutors that are socially sensitive: Politeness in educational software*. *International Journal of Human Computer Studies*, 64(1), 36-42.

Miller, C. A. (2004). *Human-Computer Etiquette: Managing expectations with intentional agents*. *Communications of the ACM*, 47(4), 31-34.

Nass, C. (2004). *Etiquette Equality: Exhibitions and expectations of computer politeness*. *Communications of the ACM*, 47(4), 35-37.

- Pallatto, J. (2007). Monthly Microsoft Patch Hides Tricky IE 7 Download. eWeek.com January 22. from <http://www.eweek.com/article2/0,1895,2086423,00.asp>
- Power, R. (2000). Tangled Web: Tales of digital crime from the shadows of cyberspace. Indianapolis: QUE Corporation.
- Pratley, C. (2004). Chris_Pratley's OneNote WebLog. 2004, from http://weblogs.asp.net/chris_pratley/archive/2004/05/05/126888.aspx
- Raskin, J. (2000). The Humane Interface. Boston: Addison-Wesley.
- Rawls, J. (2001). Justice as Fairness. Cambridge, MA: Harvard University Press.
- Rose, G., Khoo, H., & Straub, D. (1999). Current Technological Impediments to Business-to-Consumer Electronic Commerce. Communications of the AIS, I(5).
- Shectman, N., & Horowitz, L. M. (2003). Media inequality in conversation: How people behave differently when interacting with computers and people. Paper presented at the CHI (Computer Human Interaction) 2003, Ft Lauderdale, Florida.
- Transparency-International. (2001). Corruption Perceptions. www.transparency.org. 2002, from www.transparency.org
- Tzeng, J. (2004). Toward a more civilized design: studying the effects of computers that apologize. International Journal of Human-Computer Studies, 61(3), 319-345.
- Whitworth, B. (2005). Polite Computing. Behaviour & Information Technology, 24(5, September), <http://brianwhitworth.com/polite05.pdf>, 353 – 363.
- Whitworth, B. (2008). Some implications of Comparing Human and Computer Processing. Paper presented at the Proceedings of the 41st Hawaii International Conference on System Sciences.
- Whitworth, B., Aldo de Moor, A. and Liu, T. (2006). Towards a Theory of Online Social Rights. In Z. T. R. Meersman, P. Herrero et al. (Ed.), OTM Workshops 2006, LNCS 4277 (pp. 247 - 256). Berlin Heidelberg: Springer-Verlag
- Whitworth, B., & deMoor, A. (2003). Legitimate by design: Towards trusted virtual community environments. Behaviour & Information Technology, 22(1), 31-51.
- Whitworth, B., & Whitworth, E. (2004). Reducing spam by closing the social-technical gap. Computer (October), 38-45.
- Wright, R. (2001). Nonzero: The logic of human destiny. New York: Vintage Books.

Brian Whitworth is a senior lecturer at Massey University (Albany), Auckland, New Zealand. He holds a BSc (mathematics), BA (psychology), MA (neuro-psychology), and an information systems PhD He has published in journals like Small Group Research, Group Decision & Negotiation, The Database for Advances in IS, Communications of the AIS, IEEE Computer, Behavior and Information Technology (BIT), Communications of the ACM and IEEE Transactions on Systems, Man and Cybernetics. Topics include generating online agreement, voting before discussing, legitimate by design, spam and the socio-technical gap and the web of system performance. He with Aldo de Moor edits the Handbook of Research on Socio-Technical Design and Social Networking Systems (March 2009).