Politeness as a Social Computing Requirement

Brian Whitworth

Tong Liu

Institute of Information and Mathematical Sciences, Massey University, Auckland, New Zealand

Reference: Whitworth, B. & Liu, T., 2008, Politeness as a Social Computing Requirement, Chapter XXIV, p419-436, in Luppicini, R. (Ed), 2008, *Handbook of Conversation Design for Instructional Applications*, Information Science Reference, Hershey PA

ABSTRACT

This article describes how social politeness is relevant to computer system design. As the Internet becomes more social, computers now mediate social interactions, act as social agents and serve as information assistants. To succeed in these roles computers must learn a new skill - politeness. Yet selfish software is currently a widespread problem, and politeness remains a software design "blind spot". Using an informational definition of politeness, as the giving of social choice, suggests four aspects: 1. Respect, 2. Openness, 3. Helpfulness, and 4. Remembering. Examples are given to suggest how polite computing could make human-computer interactions more pleasant, and increase software usage. In contrast, if software rudeness makes the Internet an unpleasant place to be, usage may minimize. For the Internet to recognize its social potential, software must be not only useful and usable, but also polite.

INTRODUCTION

Social computing

Computers today are no longer just tools that respond passively to directions or input. 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. Perhaps this is why people often react to computers as they would to a person, even though they know it is not (Reeves & Nass, 1996). 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). Software it seems, with its ability to make choices, has crossed the threshold from inert machine to interaction participant, as the term human-computer interaction (HCI) implies. 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.

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, Alsio, & 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.

We take a *social agent* as 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 if it interacts with the customer on behalf of the company. The interaction is social even if the social agent is a computer, and an install creates a social contract even though the software is not a social entity itself. In the special case where a software agent is working for the party it is interacting with, it is a software *assistant*, working both for the user, and to the user. In such cases of human-computer interaction (HCI), social concepts like politeness apply.

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). Such behaviour is probably not illegal, but 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 (first author) decided to uninstall it. As in "The Apprentice" TV show, one reaction to assistants that don't do what you want is: "You're fired!"

Selfish software is why after 2-3 years Windows becomes "old". With computer use, the Windows taskbar soon fills with icons, each an application that finds itself important enough to load at start-up and run continuously. Such applications always load, even if you never use them, e.g. I *never* use Windows messenger but it *always* loads itself onto my taskbar. When many applications do this, it slows down the computer considerably, and taskbar icon growth is just the tip of the iceberg of what is happening to the entire computer. 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. Giving selfish applications too much freedom degrades performance until eventually only reinstalling the entire operating system can recover system performance.

Polite computing

Polite computing is about how software design can support HCI politeness. It is not about how people should be polite to people online, which various "online etiquette" guides cover. This article aims to define, specify and illustrate an information vision of polite computing.

Politeness is distinct from both usefulness and usability requirements. Usefulness addresses a system's functionality, while usability concerns how people use that functionality. The first focuses on what the computer does, and the second on how the user gets the

computer to do that. Polite computing however is not about what the computer does, nor how one can get better get it to do it. It is about social relations rather than computer power or cognitive ease. It enables software that "plays well" in a social setting, and encourages users to do the same. It addresses the requirements for social entities to, and enables better social collaboration, rather than better tool use. The contexts differ, so software could be easy to use yet rude, or polite but hard to use. While usability reduces training and documentation costs, only politeness lets a software agent work with a competent user without frustration. Both usability and politeness however fall under the rubric human-centred design.

BACKGROUND

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

"*… behaviour that is respectful or considerate to others*".

Considering and respecting others, a critical success factor in physical society, is equally relevant to online society. The predicted effect of polite computing is better human-computer interactions. While one may mistrust a polite door-to-door salesman as much as an impolite one, the polite one will get more "air time" because interacting with them is more pleasant. If politeness makes social interaction more pleasant, a polite society is a nicer place to be than an impolite one, and its people will be more willing to interact beneficially with others. Polite computing can contribute to computing by:

1. Increasing legitimate interactions.
2. Reducing anti-social attacks.
3. Increasing synergistic trade.
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 supports legitimate interactions

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). Conversely societies where win-lose corruption and conflicts still reign are among the poorest in the world (Transparency-International, 2001). Legitimate interactions offer all parties a fair choice, and are in the public good, while anti-social interactions, like theft or murder, give the "victim" little choice, and harm society overall. In contrast, polite acts are *more than fair*. 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 1). In this view politeness increases social "health", just as criminality poisons it.

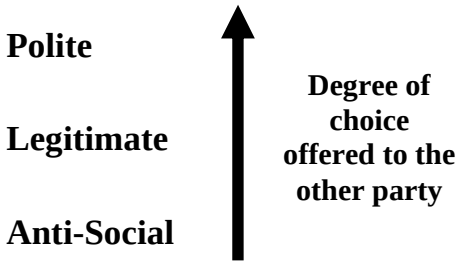


Figure 1. The social choice dimension

Politeness reduces anti-social attacks

Polite computing may have value, but shouldn't it take a back seat to security issues? Is politeness relevant if we are under attack? Yet upgrading security every time an attack exploits another loophole is a never-ending cycle. An alternative is to develop strategies to reduce motivation to attack (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). Often 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 those neutral to society to copy, or those against society to become neutral. Politeness and security seem 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.

Politeness increases 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 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 many people give to each other, they gain more than is possible by selfish activity. The success of the Open Source Software (OSS) movement illustrates this, as open source products like Linux now compete with commercial products like Windows. 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 illustrates social synergy, as we each only "sow" a small part of it, but from it can "reap" the world's knowledge interactions.

Politeness increases software use

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). Brusque and often incomprehensible error messages like the "*HTTP 404 – File not Found*" response to an unavailable web page, can imply a user fault, while a message like: "*Sorry I could not find file xxxxx.*" does not. Accusatory error messages can rub users up the wrong way, especially if it is a software error in the first place.

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

Reinventing politeness online

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. Given 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 2 distinguishes the goals of Figure 1 from their specific implementations. In this view, while each society may "implement" a different etiquette, politeness remains the common "design goal", just as legitimacy is the "spirit" behind laws that vary in detail between societies.

If politeness can take different forms in different societies, to ask which implementation applies online is to ask the wrong question. The right question is how to "reinvent" politeness in each specific online case, whether for chat, wiki, email or other groupware. Just as each different physical society develop local etiquettes and laws, so different applications may need a different politeness implementation, based on a general design "pattern", specifying politeness in information terms (Alexander, 1964).

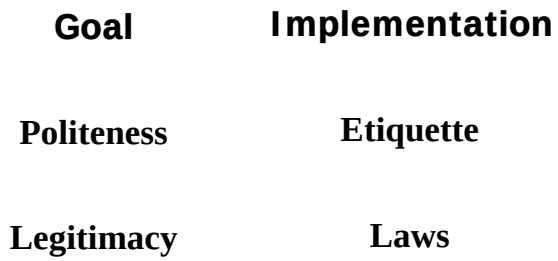


Figure 2. Social goal vs implementation

Informational politeness

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, 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.

An information definition of politeness is:

> *"… 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. "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.

Other definitions

Some define politeness as "being nice" to the other party (Nass, 2004), and argue 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 different from being kind, e.g. kind parents may not give an infant many choices, but politeness does not apply to young children who are considered to not yet know what they really want. Do software creators consider software users to be like little children, unable yet to exercise choice properly? While inexperienced users may happily let software do as it thinks is best, when children grow up they want more choice (as teenagers illustrate). The view that "software knows best" is hard to justify for the majority of today's computer-literate users. Perhaps once computer users were child-like, but today they want respect and choices from their software.

Impolite computing

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 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. Users don't like this, so many browsers prevent pop-ups. Impolite computer programs can:

1. *Use your computer's services.* Software can use your hard drive to store information cookies, or your long distance phone service for downloads.
2. *Change your computer settings.* Like browser home page, email preferences or file associations.
3. *Spy on what you do online.* Spyware, stealthware or software back doors that gather information from your computer without your knowledge, or record your mouse clicks as you surf the web and, even worse, exchange your private information with 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 : , 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, they next found that application incompatibility effectively cut them off from the Internet***."** (Pallatto, 2007)

Security cannot defend against people one invites in, especially if it is the security system taking advantage! However in a connected and free 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 old societies banishment was often considered worse than a death sentence. Likewise what online users think of a company that creates a software agent can directly impact sales. A reputation for riding roughshod over computer user's rights is not good for business.

SPECIFYING SOFTWARE AGENT POLITENESS

The widespread problem of software that is rude, inconsiderate or selfish is a general software design "blind spot" (Cooper, 1999). The specification of politeness in information terms is in its infancy, but previous work (Whitworth, 2005) suggests polite software should:

1. ***Respect the other's rights****. Polite software respects the user, does not pre-empt user choices, and does not act on or copy information without its owner's permission.*
2. ***Openly declare itself.*** *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. ***Help the other party.*** *Polite software helps users make informed choices, giving useful and understandable information when needed.*
4. ***Remember the interaction****. Polite software remembers past user choices in future interactions.*

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 HCI 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 the software but not edit, copy or distribute it. Such rights can be specified as privileges, in terms of specified information actors, methods and objects (Table 1). 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.

A respectful assistant does not interrupt unnecessarily, while selfish software, like a spoilt child, repeatedly does, 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 small child 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 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". This of course completely spoils the sample document impression, and even worse, this 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? Again what counts is what the software needs, not what the user needs.

Table 1. Social-technical Actors, Objects and Methods

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

Open

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.

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. Source data could be optional, making it backward compatible. Applications need not disclose themselves, but users will prefer sources that do. Letting users know the actions of their computer's inhabitants could help the marketplace create more polite software.

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 understood. 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 cannot understand them, is also impolite. Installing software can be complex, but so is installing satellite TV technology. In both cases users expect to hear their choices in an understandable way. 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

Finally, 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 not, put me back where I was last time." Yet most software cannot even remember what we were doing last time we met. 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.

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 the other party to again and again give the same choice reply, e.g. uploading a batch of files creates a series of overwrite questions, and software that continually asks "Overwrite Y/N?" forces the user to continuously reply "Yes". Hence most copy software also offers the "Yes to All" *meta-choice*, that remembers for the choice set. *Offering choices about choices* (meta-choices) reduces information overload, as users need only set repeated access permissions once, e.g.:

1. Always accept
2. Always reject
3. Let me choose

A general meta-choice console (GMCC) would give users a common place to see or set all meta-choices (Whitworth, 2005).

IMPLEMENTATION CASES

The impolite effect

In HCI interactions, impoliteness can cause a social failure every bit as damaging as a logic failure, e.g. the first author's new 2006 computer came with McAfee Spamkiller, which when activated overwrote my Outlook Express mail server account name and password with its own values. When checking why I could no longer receive mail, I retyped in my mail server account details, and fixed the problem. 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 this 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! I (first author) concluded it was selfish software, and uninstalled it.

Interaction situations

Other human computer interactions where politeness applies include:

1. *Errors*. Polite error messages say *we* have an error rather than *you* have an error. While computers tend to take charge when things go well, when they go wrong software seems to universally agree that the user is in fact "in charge". To ask what "we" (rather than you) want to do about an error implies the computer should also suggest solution options. Studies of users in human-computer tutorials show significant differences based on how politely the computer addresses the user, i.e. users respond differently to "Click the Enter button" vs. "Lets click the Enter button" (Mayer, Johnson, Shaw, & Sandhu, 2006).
2. *Advice and Notifications*. To interrupt impolitely disturbs the user's train of thought. For complex work, like programming, even short interruptions can cause a mental "core dump", as the user drops one thing to attend to another. The real 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 wonders: "*If you think the Assistant idea was bad, why exactly*?" (Pratley, 2004). To answer simply, he was impolite, and in XP is replaced by polite smart tags.
3. *Action requests*. Asking permission is polite because it gives the other choice, and does not pre-emptively act on a common resource, c.f. a zip extract product that puts the files it extracts as icons on the desktop, without asking! Such software tends to be used only once.
4. *Information requests.* If software asks for and gets choices from a user, it should remember them. Polite people don't ask "What is your name?" every time they meet, yet software often has no interaction memory whatsoever, 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 has no interaction memory.
5. *Installations*. 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.

Online learning

Online learning software, like WebCT or Blackboard, illustrates how politeness issues vary with channel type. While channel *richness* (rich vs. lean) was once thought the main property of computer-mediated communication (Daft & Lengel, 1986), channel properties like *linkage* (one-to-one, one-to-few or one-to-many*)* and *interactivity* (one-way or two-way) now also seem relevant (Whitworth, Gallupe, & McQueen, 2001). For example *instructor-student* online communications, like email, text messaging, chat, podcasts, cell phone or video-computer interaction, are usually *one-to-one* and *two-way*. In contrast, *instructor-class* communications are *one-to-many* and *one-way*. The rich-lean dimension is orthogonal to this distinction, e.g. an instructor can post lean text assignments, graphical lecture slides, or rich video-lessons. Email still plays a major role in online learning, though it remains largely plain text, because it is interactive. Online learning system's email and chat functions unnecessarily

duplicate existing email services, like Hotmail. Having a separate email for each class taken or taught requires students or instructors to check each class email, in addition to their normal email. For online learning systems to create normal email lists would be much more user considerate, as then students would only have to check their normal email.

In 1:1 two-way communications, like email, "the conversation channel" is the shared resource. Yet while physical society recognizes the joint ownership of communication channels, and offers everyone the right not to interact (e.g. to remain silent, to not receive junk mail, to not answer the phone, etc), the core email system gives all senders the right to put any message into any receiver's inbox. This unfairly gives all rights to the sender and none to the receiver, and enables the ongoing spam epidemic that plagues us all.

A more fundamental problem with email in online learning is that one-to-one teacher-student interactions do not scale well (Berners-Lee, 2000). While one can as easily post lessons to a large class as to a small one, handling emails for classes over 50 can be difficult. The legitimacy baseline is that students have paid for class tuition, not one-to-one on-demand tuition. Experienced instructors often restrict the use of email to personal requirements, like arranging meetings. They discourage its use for course content, e.g. "*Sorry I could not make the last class, what did I miss*?" is a real student email that I discouraged. Politeness in an interaction works two-ways, so training students to be email polite is a valid learning goal, e.g. polite emails are:

1. *Signed*. Give your name clearly – emails from nicknames like "fly-with-wind" are often unanswered.
2. *Understandable*. Give course/class number in the email title so the instructor knows the context.
3. *Personal*. Use personal email for personal issues, not issues that affect the entire class, e.g. an online instructor may paste a "When is the exam?" email into an online discussion board and answer it there, so other students can see the answer.

Class to instructor interactions, like an online assignment submission box, illustrate many-to-one one-way communication. For multi-choice quizzes the computer can also grade the submissions and give student feedback. This is scalable as the computer can handle any class size, and can remember previous tests, telling the student if he/she is improving or not. However while online exams don't need politeness, as students must take them, voluntarily online learning is a different matter. The distinction is:

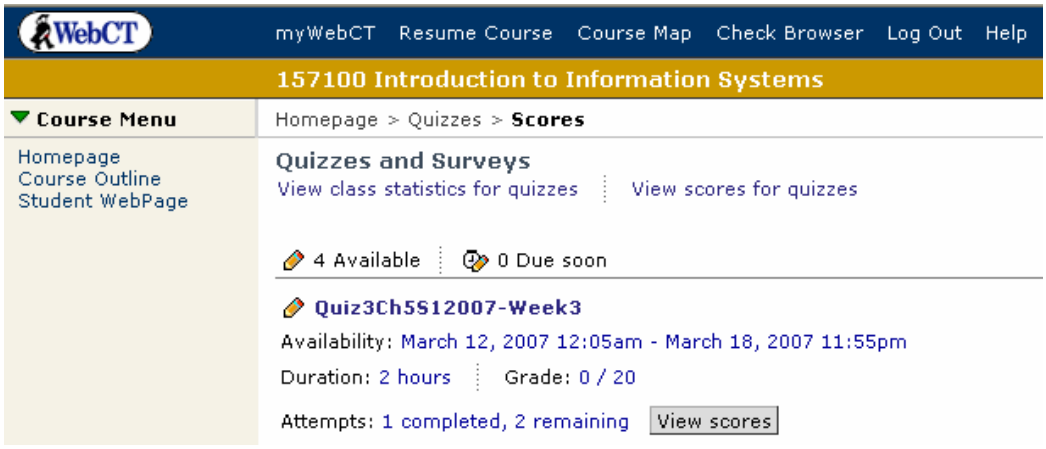
a. *Formal testing quizzes.* Usually begin and end at a fixed time, shuffle questions and options to prevent cheating, and give little content feedback. Being mandatory, politeness applies only minimally.
b. *Informal learning quizzes.* Offer choices like pausing to restart later, optional tips, answer feedback and choice of difficulty level. Being voluntary, politeness can help involve the student in the learning process.

If learning means changing ones own processing, a case can be made that all learning is voluntary. If so, polite interaction may help engage students in voluntarily online learning. The difference between a forced online quiz and an online learning experience may be politeness and respect. Online quizzes can support face-to-face lessons, e.g. if students answer online questions on a textbook chapter the week before lecture. This questioning encourages them to actively find information from the textbook, and prepares them for the weekly face-to-face class. Unlike a testing quiz, which is given *after* the class, and is graded by percentage correct, a "learning participation" quiz occurs *before* the taught class, and any reasonable

participation (e.g. 30+%) gets full points. However the quiz must be done in the week stated, and there are no "resits" for weekly participations. The quiz answers are not released until the week finishes, and students can do or redo the quiz any time in the given week. In practice, those who do poorly in testing quizzes also tend to omit the learning quizzes. However the good students find them an excellent way to learn.

Most online learning systems seem designed to give information to teachers rather than students, who get learning feedback only with difficulty, e.g. Figure 3a shows a "View Scores" button which when clicked gives Figure 3bthat shows a score. Few students then realize that clicking the underlined "1" gives feedback on the right answers. While online teachers can "see" everything, like when and for how long students are online, students struggle to see what could help them learn in online software.

a.



b.

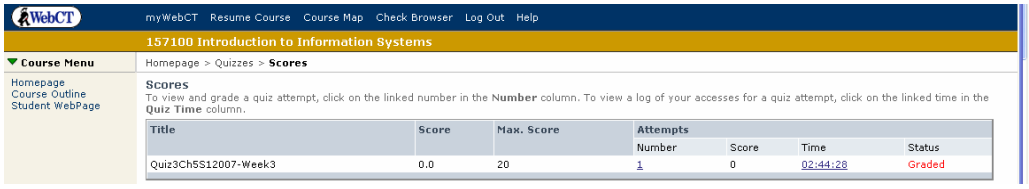


Figure 3. Getting quiz feedback in WebCT

Class-to-class FAQ boards, where students answer each other's questions, are many-to-many, two-way interactions that scale well to all class sizes. Respecting and using class member knowledge is not only popular with students, but for fast changing subjects, like web-programming, almost essential. If young people learn mainly from their peers, involving their peers in online learning seems sensible, and polite computing could enable this.

Polite computing suggests voluntary choice is a new online learning dimension. Its application however requires a complete redesign of current teacher focused systems like WebCT. The online classroom must move from what is essentially a software supported dictatorship to a system that invites voluntary student participation, based on a balance of rights and choices.

FUTURE TRENDS

Polite computing suggests computers will increasingly:

1. Remember interaction data rather than object data.
2. Become human assistants or agents rather than independent actors.

3. Support politeness rather than selfishness in online interaction.

Remember the interaction

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 cannot relate to people. To spy on users at every opportunity 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. Most software today is in the same category, and modern airport toilets seem more aware of their users than the average personal computer. Hopefully tomorrow's software will make HCI memory its business, as its primary role will be to work for people, not for itself.

Computers as assistants or agents

There are several reasons why people should control computers, not the reverse. *Firstly*, while computers manage vast amounts of data with ease, they handle context changes poorly, and outside their fixed parameters can seem very stupid. So-called "smart" computing (Kurzweil, 1999) usually 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 impolite software is not inconceivable.

Fortunately the future of computers probably 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. These arguments suggest that if the role of computers is to assist, they should learn to be polite.

Online politeness will grow

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 include third party blockers, cleaners and filters, of various sorts, whose main aim is to put users back in charge of their computer estate. Such applications are the most frequent accesses at Internet download sites. Like all wars, if software declares war on user choice, everyone will lose in the long run. If the Internet is a battlefield, no-one will want to go there. Some compare the Internet to the U.S. Wild West, and others talk of the "hunter-gatherers of the information age" (Meyrowitz, 1985, p315). Yet the Stone Age and the U.S. Wild West evolved into civil society, and so perhaps it is time to introduce civility to the Internet. What took physical society thousands of years may occur online in only a few years e.g. Wikipedia began with few rules and one leader, but now to combat "trolls" who trash data, has many rules (including copyright) and many roles, like "Steward", "Bureaucrat" and "Sysop" (Whitworth, Aldo de Moor, & Liu, 2006). Yet the real force behind Wikipedia is the majority's enjoyment of working together considerately, not its ability to deal with the anti-social minority.

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 selling, but improving the customer relationship that leads to sales. By giving customers choice, 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 is an online way of life.

FUTURE RESEARCH

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 its privacy. The ability of software to hold users hostage to its power may be declining. Where customers choose their software, we make a simple prediction: *Polite software will be used more and deleted or disabled less than impolite software*.

An experimental test of polite computing *value* requires a comparison of polite versus impolite applications on measures like willingness to use, attitude to the software, willingness to purchase and user satisfaction. Politeness here is defined to apply not just to language, conversations, or people, but also to human-computer interactions. Research can show if computer users really value politeness in HCI interactions like application installations, user help, online learning, email, messaging and bulletin boards, to mention a few. This politeness is not just the words used, but also the software actions taken. The relative value of the proposed politeness sub-aspects (respect, openness, helpfulness and remembering) can also be compared. Correlational studies could compare rated application politeness with market success. Longitudinal studies could determine if successful applications become more polite over time. Ethnographic studies could explore how users perceive polite and impolite software.

The *scope* of online politeness also bears investigation. Our definition implies that young or inexperienced users will tolerate impolite agents like Mr Clippy more than experienced users. Also it has been proposed that for interactions mandated by law, or other coerced acts, politeness will apply less. Other individual differences, including gender, age and culture, may also mediate the user reaction to impolite software. Cultural differences in polite computing raise highly complex issues of roles and social structures, and may affect the boundary between what is required and what is polite.

CONCLUSIONS

Polite software asks before it allocates computer resources, openly declares itself and its acts, does not unnecessarily interrupt or draw attention to itself, offers understandable choices, and remembers past interactions. Conversely, impolite software acts without asking, does things secretly, interrupts unnecessarily, offers confusing choices, and has no recall of its past interactions with you.

If polite software attracts users, impolite software can drive them away. This implies a new type of IS error – *social error*. A program syntax error fails to support the needs of the computer technology. A software usability error fails to support the psychological needs of

the computer user. However a social error means the software fails to support the equally critical needs of human social interaction. While users misunderstand systems designed with poor usability, they understand impolite software all too well, and that is why they walk away from the interaction. Whether a system fails because the computer can't run it, the user can't run it, or the user *won't* run it, makes no difference. The end effect is still that the *application doesn't run*. A software social error gives the same outcome as a software crash or user failure. Indeed social errors may be even worse, as it is in the nature of people to actively seek retribution against those who wrong others in social interactions.

A future is envisaged where software politeness is a critical requirement for social-technical system success, especially where user willingness to participate counts. Polite computing could be taught in system design classes, along with other system requirements. A "politeness seal" could credit applications that give rather than take user choice. If physical society in general sees the value of politeness, online society should follow that lead. As software becomes not only useful and usable but also polite, the Internet may at last recognize its social potential.

ACKNOWLEDGEMENTS

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

REFERENCES

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

Berners-Lee, T. (2000). *Weaving The Web: The original design and ultimate destiny of the world wide web*. New York: Harper-Collins.

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

Daft, R. L., & Lengel, R. H. (1986). Organizational information requirements, media richness and structural design. *Management Science, 32*(5, May), 554-571.

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

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

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

Gauze, C. F. (2003). *I See You're Writing an Article...* (March). INK19. Available: 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.

Kurzweil, R. (1999). *The Age of Spiritual Machines*. Toronto: Penguin Books.

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.

Meyrowitz, J. (1985). *No Sense of Place: The impact of electronic media on social behavior*. New York: Oxford University Press.

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* (January 22). Available: http://www.eweek.com/article2/0,1895,2086423,00.asp.
PCMagazine. (2001). *20th Anniversary of the PC Survey Results*. Available: http://www.pcmag.com/article2/0,1759,57454,00.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*. Available: 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.
Reeves, B., & Nass, C. (1996). *The Media Equation: How people treat computers, television, and new media like real people and places*. New York: Cambridge University Press/ICSLI.
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). Available: 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,* (5, September, http://brianwhitworth.com/polite05.pdf), 353 – 363.
Whitworth, B. (2006). Spam as a symptom of electronic communication technologies that ignore social requirements. In C. Ghaoui (Ed.), *Encyclopaedia of Human Computer Interaction* (pp. 559-566). London: Idea Group Reference.
Whitworth, B., Aldo de Moor, & Liu, T. (2006, Nov 2 - Nov 3). *Towards a Theory of Online Social Rights* Paper presented at the International Workshop on Community Informatics (COMINF'06), Montpellier, France.
Whitworth, B., & deMoor, A. (2003). Legitimate by design: Towards trusted virtual community environments. *Behaviour & Information Technology, 22*(1), 31-51.
Whitworth, B., Gallupe, B., & McQueen, R. (2001). Generating agreement in computer-mediated groups. *Small Group Research, 32*(5), 621-661.
Whitworth, B., & Whitworth, E. (2004). Reducing spam by closing the social-technical gap. *Computer* (October), 38-45. http://brianwhitworth.com/papers.html
Wright, R. (2001). *Nonzero: The logic of human destiny*. New York: Vintage Books.