Chapter XIV
Politeness as a Social Computing Requirement

Brian Whitworth
Massey University, New Zealand

Tong Liu
Massey University, New Zealand

ABSTRACT

This chapter 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 e-mail, 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 interacted 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 do not treat computers as people outside the mediation context (Goldstein, Alsio, & Werdenhoff, 2002)—just as people do not 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 taken as an interacting entity that represents another social entity in an interaction, either person or group, for example, 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, for example, 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 do not 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, for example, I *never* use Windows messenger but it *always* loads itself onto my taskbar. When many applications do this, it slows down the computer