Cite as:
Greenberg, S., Witten, I. H., and Finlay, J. (1993). Software personalization. In A. Ralston and E. D. Reilly (Eds.), Encyclopedia of Computer Science, pp. 1240-1241. Van Nostrand Reinhold, New York.

SOFTWARE PERSONALIZATION

Index references: human-computer interaction, user modeling, personalization, intelligent tutoring systems, intelligent help systems, adaptive systems.

Software is *personalizable* if its behaviour can be altered to match a particular individual's needs. Since a good user interface will make a computer program as convenient to use as possible, it should have the means to be adapted or personalized to a user's needs. Suppose that two users are using the same application program at identical workstations, doing identical things. Depending upon personal needs and tastes, they may require slightly or even completely different styles of user interface. For instance, if one has good eyesight and the other has poor eyesight, the former may want text displayed in a smaller than average font to get more on the display, whereas the other may well want a large font to make the display more legible. Another example is an intelligent help system that customizes the content and presentation of material to a particular person. The importance of personalization stems from the need for mass-produced software to accommodate the individual differences typically found within a large but diverse clientele of users.

Software personalization has its roots in the operating systems of the late 1960's and 1970's that provided simple yet effective ways for programmers to customize their command-based computing environment. Users, for example, could replace oft-used but unwieldy command lines with simpler abbreviations and aliases. Similarly, preferences and options that override system defaults could be maintained by setting environment variables in a database referred to by a program at run time. But the major breakthrough in the design of personalizable operating system interfaces was to eliminate all differences between invoking a system program and a user program (as in Unix). This is significant because it allows one to tailor and extend a system to individual needs simply by writing utility programs (programs or scripts of command line sequences) and putting them in the right place, without having to alter the innards of the system in any way.

Some of these ideas have migrated to modern-day graphical computing environments that are oriented towards the non-programmer. Instead of personalizing the environment through setting syntactically baroque options or writing short programs, users are presented with a *property sheet*—a window-based form—that displays items and their options as selectable and modifiable fields. For example, the Apple Macintosh provides

users with a control panel that adjusts the "look and feel" of the user interface. Through it, users can choose the pattern of the background screen, adjust the repeat rate of the keyboard, tune the sensitivity of double mouse clicks, and so on.

A significant development in personalizable software is the notion of *user modeling,* a technique for building and maintaining a model of the user based upon information presented to the system, and then applying the model to a particular use. The key point is that the design of the personalizable software is divided into several roles:

1. The *modeling agent* selects and/or constructs an appropriate user model. The agent can be either a person (the user or designer) or the system itself. For example, users acts as modeling agents when they explicitly adjust preferences in a property sheet. Alternatively, the system may automatically build a user model by inferring key characteristics from the user's input.
2. The *user model* contains information about an individual user. The information represented by the model can be quite varied. One example is a simple scalar value that indicates the user's level of expertise. Other instances are profiles where the user is characterised as a series of weighted parameters mapped onto weighted tasks, and input traces where the model maintains a record of a user's important activities.
3. The *model use* governs how the model is to be used. Examples include predicting and anticipating user actions, adjusting the interface look and feel, and so on. The method for using the model can be hard-wired into the software code, or maintained explicitly as a set of independent and changable rules within an expert system.

Good examples of the application of user modeling techniques to personalizable software can be found in *intelligent tutoring systems*. These not only know what to teach, but can customize how they will present the material according to whois being taught. Based upon input from the learner, the system monitors and models the learner's knowledge, compares it with its own knowledge base, and directs tuition towards those parts that are identified as missing from the learner's understanding. Similarly, *intelligent help systems* provide users with help appropriate to their problem and level of knowledge. They can also expand the user's knowledge by suggesting new approaches and by correcting erroneous inputs. The user model may therefore incorporate an assessment of the user's expertise in a particular task, a model of the task in hand, and a model of the concepts known to the user.

Software personalization is still in its infancy. While it has been clearly accepted in its simpler forms, such as the ability of users to control explicitly a limited set of attributes,

intelligent systems that automatically adapt their behaviour to the user are rare outside research institutes. However, the future is promising, for personalizable software is a practical and economic way of allowing systems to conform to the user, rather than forcing users to conform to the system.

References

1985. Greenberg, S. and Witten, I.H. *Adaptive personalized interfaces—A question of viability*. Behaviour and Information Technology, **4**(1), pp 31-45.

1991. Kobsa, A. (Editor). Journal of User Modeling and User-Adapted Interaction. Kluwer academic publishers, Forthcoming.

1983. Rich, E. Users are individuals: Individualizing user models. International Journal of Man Machine Studies, **18**(3), 199-214.

1982. Sleeman, D. and Brown, J.S. *Intelligent Tutoring Systems*. Academic Press, London.

Saul Greenberg and Ian H. Witten
Department of Computer Science
University of Calgary,
Calgary, Canada T2N 1N4

Janet Finlay
Department of Computer Science
University of York,
York, England VO1 5DD