JAKOB NIELSEN

As They May Work

all illustrations: Teofilo Olivieri

One of the basic questions during the development of a computer system and its user interface is what the users will want to do with the system. Unfortunately, a task analysis of peoples' current activities is not sufficient to predict what they will do in the future. It is well known that peoples' use of computers change over time and that new and unexpected uses are found for most new systems. A famous historical example is the introduction of the ARPANET computer network which led to a surprising widespread use of electronic mail even though it had been primarily intended for remote logins and the sharing of computers and databases. Another example of unexpected use of a computer system is the way

some users build up fairly large databases in spreadsheet programs because they find them easier to use than regular database programs.

One way to handle this problem would be to implement the basic features for which a user need has been established and then release updated versions as additional user needs become apparent. Since software maintenance is expensive, it is better to include as large as possible a proportion of the eventual feature set in the initial software design. From an interface design perspective, a comprehensive initial feature list will result in a cleaner and more integrated design than a system that has been extensively retrofitted. Unfortunately, it is impossible to predict every feature users will ever need or want in the future with complete accuracy. Even so, it is still useful to have a partial solution that would suggest a number of features that users would likely want, given that it has already been determined that they need certain features.

This column describes a technique for extending a task analysis based on the goal composition principle suggested by Clayton Lewis ("A research agenda for the nineties in human-computer interaction," *Human-Computer Interaction,* vol. 5, no. 2, page 134). Basically, goal composition starts by considering each primary goal that the user may have when using the system. A list of possible additional features is then generated by combining each of these goals with a set of general meta-goals that extend the primary goals. These general meta-goals are called goal composition mechanisms. As a non-computer example, a goal composition mechanism for cooking would be to find a way to serve the food. Thus, the primary goal of preparing coffee can be extended to the further goal of getting a coffee cup.

For the following discussion of goal composition, electronic mail is used as an example to illustrate how each mechanism can be used to generate further potential user goals. The basic feature of email is to send a message to another user in the form of a text file. An initial task analysis might have shown that email can be useful in many cases such as when the other user is located far away or is hard to reach and that the basic user goals are sending and receiving messages.

The goal composition mechanisms described here are fairly general and will apply to most of the features of most computer systems. Four of these mechanisms were described by Clayton Lewis and an additional nine mechanisms were discovered through further analysis. It is an open research question to what degree the list is complete in the sense that no more goal composition mechanisms should be added. Additional goal composition mechanisms might be appropriate for special application domains or interaction styles, but the focus of this column is on generally applicable mechanisms.

The three main categories of goal composition are generalization, integration, and user control.

Generalization Mechanisms

Generalization allows users to increase the scope of a feature by applying it to more objects. The three generalization mechanisms are multiplexing, reuse, and supergoaling.

Multiplexing

Multiplexing refers to the ability to have many instances of a goal be achieved as a single goal. For example, when sending email, one often wants to send to many users in a single operation, so multiplexing would at least imply the ability to specify many recipients for a given message. Extended support for email multiplexing would involve features such as distribution lists to allow the user to refer to many recipients by a single name.

Reuse

Reuse enables the user to utilize part of the work towards one goal when wanting t o

achieve a similar goal. For example, a spreadsheet user who has generated a spreadsheet formula to sum one row might have the further goal of copying this formula to sum other rows. A simplistic copy and paste implementation which allowed the user to move the formula would not achieve this reuse goal since the copies of the formula would still sum the original row. Most spreadsheets therefore provide some operation for copying a formula while updating it to refer to a new set of cells.

In the email example, users should be allowed to resend an old message, while possibly changing the list of recipients and editing the text. Note that this reuse of old messages would be significantly more useful if the user had an easy way to find out what previous messages were archived and had an easy way to retrieve a specific message according to a variety of criteria. Reuse may thus often take place in combination with a recording and retrieving mechanism as further discussed below.

Supergoaling

Supergoaling involves the use of any particular goal as a subgoal for some larger activity. Even without any particular supergoaling support, most goals form part of a goal hierarchy since they are achieved for some higher purpose. Supergoaling may be supported by programmable interfaces that allows users to combine features from multiple application to a single task, or they may be supported by linking and scripting mechanisms that will allow third-party applications to access the features of the program currently under development.

Email is not just a stand-alone application but also a support mechanism for many other user goals. Email-enabled applications are currently all the rage, indicating that application vendors have discovered that email can be a subgoal for many other activities: when you have created a spreadsheet, you may want to share it with others. Similarly, a World-Wide Web browser might include a way for the user to send a hypertext link to another user with a recommendation to look it up. As another example, if we were designing a spelling checker for word processing, it might be reasonable to supergoal it for use in other contexts such as email authoring.

Integration Mechanisms

Integration mechanisms aim at allowing each individual feature to be used in combination with other system facilities. Some of these goal composition mechanisms may be supported automatically by multiprocessing window systems, but even so, a system designer may have to consider how to support them better than the basic level provided by the system. The four integration mechanisms are interleaving, suspension and postponement, result passing, and automated use.

Interleaving

Interleaving goals allows the user to accomplish other goals at the same time as the particular goal that is being analyzed. When composing email users may have the goal of checking their online calendar and this may be accomplished by allowing multiple windows to be open. Users may furthermore want to add an appointment received over email to the calendar, and this goal might be achieved by drag-and-drop if the calendar can parse the text of the appointment message. Even better, a fully object-oriented email system may know that an appointment object would need to be scheduled and would allow users to do so without calling up the calendar: upon receiving the incoming email message, the system would recognize an appointment object and would check the user's calendar to see whether the user was available or busy and would inform the user of the result when the user opened the message to read it.

Suspension and Postponement

Users should be allowed to suspend work towards goal in order to resume the activity later. When writing an email message, the user might need to find additional information or the user might just

want to procrastinate. In any case, the user should be allowed to save whatever work has already been done on the email message in order to return to it later. Most email systems at least allow users to go to other windows before completing the message (that is, the user can interleave other tasks), but the email system should also allow the user to close message windows and have several unfinished messages stored for later completion.

Even if the user finishes work on the task, the user may not want the resulting action to happen until some additional condition has been met. In that case, the user should be allowed to postpone completion of the goal until the conditions are met. For example, when writing an email message reminding people to attend a meeting, the user should be able to tell the system to send the mail an hour before the meeting is scheduled to start.

Jakob Nielsen is Distinguished Engineer in the Human Factors Engineering department of SunSoft (the software planet of Sun Microsystems).

Suspension and postponement features imply the additional need for methods of learning what unfinished goals are present in the system. For suspended goals, the user will need to either resume or delete the unfinished work, and for postponed goals, changing conditions may cause the user to revise their request for future action. For example, if the meeting was cancelled, the user would not want to have the reminder sent out.

Result Passing

Result passing involves using the result of one goal to help achieve some other goal.

In email, the goal of reading a message often gives way to the goal of sending a reply message to the originator of the first message, and many email programs therefore include a special reply function that passes the address of the originator from the reader interface to the sender interface. Reply functions often also offer the ability to include parts of the original message in the reply message for easy reference during discussions. Even though simple copy-and-paste commands can be often be used for result passing, the inclusion of a quoted message in email normally involves some slight additions to the text to indicate its status as a quotation. A practical design guideline derived from the result passing principle is an ability for programs to save their state in standard formats as far as possible so that the result of using them can be transferred to as many other programs as possible. For example, a word processor might have the ability to save documents as plain ASCII files in addition to various formats with more fancy formatting.

METHODS & TOOLS COLUMN EDITOR
Jakob Nielsen
SunSoft
2550 Garcia Ave., UMTV 19-107
Mountain View, CA 94043-1100
jakob.nielsen@sun.com

Automated Use

Automated use requires the ability to have other programs activate the new facility and use it to achieve the goal. By making this possible, many new types of integration and an additional level of flexibility become possible, since these other programs might do anything that can be programmed. If users want to achieve a given goal interactively, then this goal might presumably also be relevant for other programs used by those users. Note that automated use is closely related to the issue of supergoaling discussed under generalization mechanisms. The difference is that supergoaling involves having the users construct larger goals for their dynamically changing purposes (possibly through a macro scripting language) whereas automated use is performed by the computer without explicit human intervention.

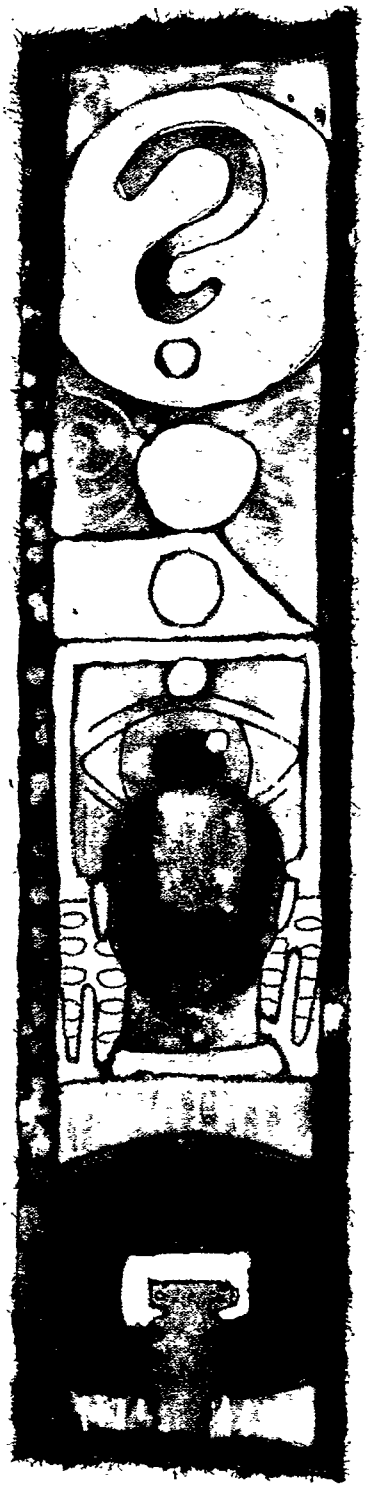
The above example of supergoaling a spelling checker to allow it to be accessed from the email program might be extended to an example of automated use where the computer would check the spelling of all email messages without being asked for it. Automated use of email would allow a third party program to generate a complete email message, complete with a header specifying the recipient's complete address. A more accommodating email facility would enable the other programs to send email using aliases, mailing lists, and any other simplifying features normally made available to an interactive user. As an example, if the license cost of a software system was dependent on how often it was used, the system could automatically email its vendor each month with information on its use. A less benevolent example is the infamous IBM Christmas card chain letter that was automatically forwarded to all users in the recipient's alias directory, and this example obviously indicates the need for some safeguards on automated use of features like email.

User Control Mechanisms

User control mechanisms allow users to inspect and change the way the computer carries out their instructions. The six user control mechanisms are monitoring, result investigation, recording and retrieving, alternative enumeration, reverting, and modification and editing.

Monitoring

After the user has asked the computer to perform a task, the user may want to determine whether the goal has been achieved or what progress is being made toward the goal. For the goal of sending email to a recipient, obvious monitoring goals would be to check whether the message has made it to the other end and whether the recipient has seen it yet.



Result Investigation

Users sometimes have difficulty in predicting exactly what a given computer command will do, and result investigation features may therefore be needed to explain the outcome of user actions to enable users to evaluate whether they actually did what they wanted to do. Of course, providing feedback is one of the more basic user interface design guidelines, but result investigation goes one step further than regular feedback and is only activated upon explicit user request. Also, result investigation might be initiated in a predictive mode before the user has committed to performing the action. For email, result investigation might allow the user to expand aliases and mailing lists to get a full list of all the recipients before sending a message.

Recording and Retrieving

In addition to doing things, users often also want to keep a record of what was accomplished. For traditional command-line interfaces, this record is clearly visible by scrolling back the window, but for most GUIs, actions are only visible while they are being carried out, so an additional feature is needed to access the interaction history. In our email example, users often want to know what mail has been sent to a given recipient or exactly who received a given message, so they need to be able to access a log of their outgoing mail. Of course, recording user actions implies the further need for mechanisms to retrieve the recorded information according to a variety of search criteria (one of which will normally be time).

Alternative Enumeration

Often, several possible user goals are very closely related and it would be difficult for the user to remember the exact difference between the desired outcomes. In these cases, the user should be able to express the goal in general terms and have the computer enumerate the available alternatives. If the user wants to send email to Joe Something, the computer should allow the user to enter "Joe" and then show a list of the seven Joes in the user's alias list.

Reverting

Users may sometimes realize that their actions resulted in undesired consequences, and they may therefore want to revert to the system state that was in effect before the goal was accomplished. This mechanism is often realized

by an undo command. For email, it may be impossible to retract statements that have already been read by the recipient, but it should certainly be possible to cancel a previously sent message if it has not been read yet.

Modification and Editing

In addition to the ability to cancel an operation and start over, users often need the ability to . If an email message has bounced due to an address error, the user should be able to resend it and edit the address without having to retype the address all over (with the possibility of new typos). Notice how a "resend" feature might be implemented by a combination of the standard features for "retrieve" (message from the record of outgoing mail) and "reuse" (this message as the basis for a new one). Even so, users might still like to have an explicit "resend" feature that would eliminate the need to search for the original message.

Another example that highlights the need for editing is mail forwarding: Users may want to add their own comments or remove irrelevant parts of the message before sending it on.

Conclusions

The table lists the goal composition mechanisms discussed in this column. The table can be used as a checklist when designing new computer features to make sure that one has considered these possible extensions of the new features. Not all composition mechanisms apply to all the goals one might want to support on a computer. Also, one should be careful not to overload an interface with too many features. Even so, the experience of designers who have used my checklist indicates that it can be useful in rounding out ideas for new features to make sure that relevant extensions have not been overlooked. As will be apparent from the examples, the features suggested by goal composition can often be designed in quite different ways. The goal composition technique can only suggest the areas to which a designer should pay attention.

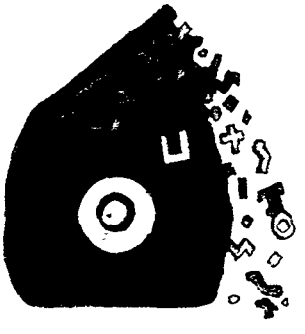


Table 1

Checklist of goal composition mechanisms.

Generalization Mechanisms

Multiplexing: *having many instances of a goal be achieved as a single goal*
Reuse: *using part of the work towards one goal when wanting to achieve a similar goal*
Supergoaling: *using any particular goal as a subgoal for some larger activity*

Integration Mechanisms

Interleaving: *accomplishing other goals at the same time*
Suspension and Postponement
Suspension and Postponement: *suspending work towards a goal in order to resume the activity later*
Result Passing: *using the result of one goal to help achieve some other goal*
Automated Use: *activating a new facility by other programs to achieve the goal*

User Control Mechanisms

Monitoring: *determining whether the goal has been achieved or what progress is being made toward the goal*
Result Investigation: *explaining the outcome of user actions to allow users to evaluate whether they actually did what they wanted to do*
Recording and Retrieving: *using a record of what was accomplished*
Alternative Enumeration: *expressing the goal in general terms and having the computer enumerate available alternatives*
Reverting: *reverting to the system state that was in effect before the goal was accomplished*
Modifying and Editing: *modifying the outcomes of prior operations*

It will then be the responsibility of the designer to determine the best way to extend the basic user goal with additional features suggested by goal composition. Furthermore, the design of real applications needs to consider the extent to which particular features will actually be used since it is often best to drop unimportant features. The goal composition technique cannot offer predictions of the relative importance and frequency of use of system features it suggests.