EVALUATION AND ANALYSIS OF USERS' ACTIVITY ORGANIZATION

Liam Bannon, Allen Cypher, Steven Greenspan, and Melissa L. Monty
Institute for Cognitive Science
University of California, San Diego

Abstract

Our analyses of the activities performed by users of computer systems show complex patterns of interleaved activities. Current human - computer interfaces provide little support for the kinds of problems users encounter when attempting to accomplish several different tasks in a single session. In this paper we develop a framework for discussing the characteristics of activities, in terms of activity structures, and provide a number of conceptual guidelines for developing an interface which supports activity coordination. The concept of a workspace is introduced as a unifying construct for reducing the mental workload when switching tasks, and for supporting contextually-driven interpretations of the users' activity structures.

This document was jointly authored by the members of the Activity Structures Research Group listed above in alphabetical order. We wish to thank Don Norman, Bob Glushko, and Jonathan Grudin for their insightful comments on earlier drafts of this paper and Tom Erickson for his related ideas and software design.

The Activity Structures Research Group operates under the auspices of the Human-Machine Interface project (UCSD), and is comprised of computer scientists and psychologists. This research is supported by Contract N00014-79-C-0323, NR 667-437 with the Personnel and Training Research Programs of the Office of Naval Research and by a grant from the System Development Foundation. Steven L. Greenspan is supported on a Postdoctoral Fellowship by Grant PHS MH 14268 to the Center for Human Information Processing from the National Institute of Mental Health.

Requests for reprints should be sent to HMI, Institute for Cognitive Science C-015; University of California, San Diego; La Jolla, California, 92093, USA.

Introduction

The intent of this paper is to give an overview of the work being carried out by the Activity Structures Research Group at UCSD. Our interests are focused upon (a) developing a methodology for studying the complex structuring of activities that often occurs in human-computer interaction, and (b) designing a human-computer interface that is supportive of the user's conception of these multilevel complex activity structures. Our observations of user-computer interactions strongly suggest that the command sequences employed by users are structured and coherent, and that users require computer systems which (a) provide information to reorient the user when resuming tasks which have been interrupted, and (b) minimize the interference incurred when setting up a new task.

We started by analyzing patterns of user-computer interaction. We did this by examining history lists of commands performed by users, aided by a system that reminded users to periodically annotate their lists with descriptions of their intentions during the session. An extract from one of these augmented history lists is provided in Figure 1. Files such as the one shown were collected automatically over a number of sessions with each subject. From the command histories (on the numbered lines) and the user's comments (in < >'s), we were able to discern the users' goals, subgoals and tasks, and how they evolved over time.

The data reveal interesting patterns of commands. Users seem to engage in a number of different activities that can be partitioned into sets of goal-related tasks. For instance, in any single session a user may edit an article, write and debug a program, search for an old file, and answer mail. Often users will work on one task and then, before completing that task, switch to a second one. Tasks tend to be nested within one another; digressions are frequent. When a record is made of these commands as they occur temporally, as in the history list, information on the tasks and goals of the users is lost. An alternative organization of user commands which preserves their task specificity would be useful (see Figure 2). Separating the history list into such functionally distinct units is difficult due to the interleaved activities of the user. In attacking this problem from an Artificial Intelligence perspective, Huff & Lesser (1982) describe a system that utilizes clues in the input patterns to infer the user's activity structures. Our alternative approach assists the user in explicitly indicating the goals associated with each command. Once the user's separate

```
< Write Info Retrieval report. Its going to take a
long time and be interrupted by other activities>
   15   vi IRreport
< interrupted to prepare a memo. Send note to gm
about outcome>
   16   snd gm
< back to IRreport>
   17   fg
   18   lf HMI
   19   lf hmi
   20   cdd hmi/IR
   21   ln /csl/norman/IRreport .
   22   ll
< Need to look over the log of old Discussion
Reports. While at it, decide to trim the log file
of irrelevant stuff that need not be kept, but that
makes file size huge>
   23   msg #hmilog
   24   ll /csl/bin/hmilog
   25   chmod 666 /csl/bin/hmilog
   26   update
< doing Update reminded me of my goals. Decide to
do the memo >
   27   snd der gm smith
   28   j
   29   lf
   30   cdh
   31   ma
   32   f
< the "f" (finger) showed me funny listing for
smith's address., so ...>
   33   grep smith /etc/passwd
< ok, back to work writing the IR report>
   34   cdpop
   35   lf
   36   vi IR*
< start spelling correction in background>
   37   correct IRreport
   38   bg
< while waiting, read mail and play with numeric
operations in cshell -- triggered by my reading
of the history scripts that S.D. developed>
   39   msg
   40   ps
   41   @ n = 5 + 3
   42   e $n
   43   @n = n * 123
```

Figure 1. Example of an annotated history list showing complex interleaving of activities.

tasks are known, it is possible to treat each task as a separate workspace. The user's pattern of activity may then be characterized as movements from one workspace to another.

The workspace

Our framework for discussing activity structures rests heavily on the notion of a "workspace": an environment dedicated to allowing easy user manipulation of activities to achieve a particular goal or set of functionally related goals. Making the workspace the basic entity in activities coordination has important implications at two levels. From the *users'* perspective, workspaces must have highly dynamic internal structures which can be modified as users reformulate their goals. From the *system's* perspective, a workspace contains tools and data relevant to the users' goals and, in addition, provides a record of the ongoing activities or processes resulting from applying a set of tools to a set of data structures. The internal structure of the workspace thus reflects both the users' goals and the software tools that the computer system can provide for accomplishing these goals.

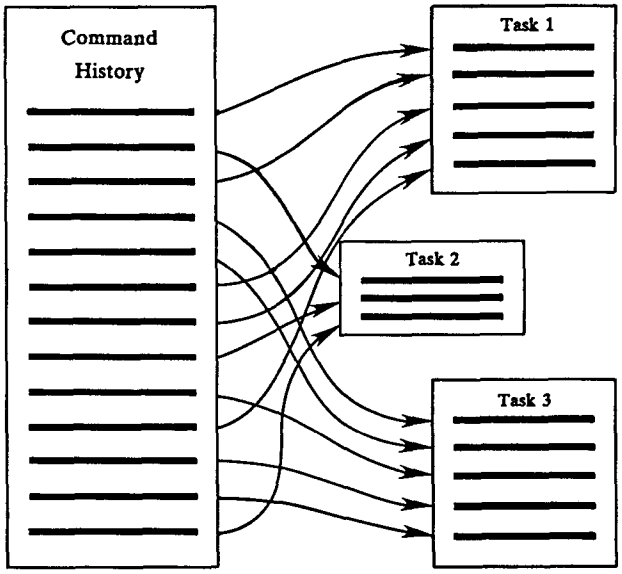

Figure 2. Conceptual model illustrating the reorganization of the command history into funtionally distinct task related command sets.

Workspaces combine the ideas behind functionally-defined *directories* and *workbenches*, [1] but they are more dynamic — they preserve information about the status of activities, and they are capable of being partitioned, recombined, and interrelated by the user. Personal notes and comments on the various files together with a system for displaying information about the workspace organization can be strategically located within the workspace, functioning as memory aids and descriptors of the contents or goals defined by the workspace.

Users encountering a computer system for the first time need models for organizing their activities and accomplishing tasks. They could be provided with a number of "skeletal" workspaces containing a set of tools and examples of their use. Users would be encouraged to build upon the system-provided frameworks by rearranging and personalizing them to suit their needs.

---

1. A workbench provides an environment which contains a set of programs that are functionally related to a particular type of task (e.g., "writing aids", or "programming").

Activity Coordination Issues

Our investigations into activity structuring resulted in the identification of several classes of activity coordination problems. Based on our analyses of users' annotated command histories, we are suggesting some guidelines for the development of a uniform interface system which should handle the coordination and execution of activities at many levels.

Reducing mental load when switching tasks

A common problem with many interfaces is how to avoid situations where users must struggle with the interface whenever they wish to switch topics. Suppose a user is doing some task but then thinks of an idea relevant to another task. In order for the user to be able to make a note about an idea in the relevant environment, the user must stop the current task, switch directories, open a new file, and, if the idea is still remembered after all this activity, finally type it in. Many users cope with this problem by keeping a pad of paper handy for such occasions. Our proposals are designed to decrease the amount of cognitive overhead — the load on working memory — required for the system interface and to free the user's thinking capacity for the true tasks at hand.

From the user's point of view, entering text should require practically no effort. This implies that naming, organizing, and locating the text within the file system (tasks intended to aid later retrieval) should not impede the activity of entering the text. If necessary, these activities should be delayed indefinitely. The workspace should allow the users to place notes at any position in the current workspace, just as they might "jot down" a note in the margin of a paper. Memory for the context within which the note was created will function as a natural aid to retrieval of the note long after it was written. When the user is ready to devote time to the task, the note could be named, duplicated, and/or relocated. The system would also allow the user to insert reminders and identifiers within the workspace itself, for quick reference on the purpose or context of that workspace.

Suspending and resuming activities

Our observation that users rarely complete any time-consuming activity before beginning another task suggests that a mechanism for easily suspending the current activity is essential. For example, we have found that while doing a task, users often encounter some problem that should be fixed immediately. This means stopping the main task, doing the fix or housekeeping chore, and then resuming the initial task. These contextually-driven activities we call "digressions". While they are common, and often a means of accomplishing many secondary goals which might be forgotten otherwise, they can distract a user from the goals of the main task.

The workspace system should support digressions while providing a large-scale place holder to facilitate easy return to previous activities. Restarting an activity should return the user to the precise state and location within the environment which was frozen, thereby freeing users from the difficult task of remembering what they were doing previously. Some systems currently support this. Berkeley Unix allows tasks to be stopped and resumed, but it does not readily support retention of the full context. Some of the new computer systems with "window" facilities, such as the Xerox STAR (Smith, Irby, Kimball & Verplank, 1982), support certain digressions by allowing users to maintain multiple windows on the screen and to enlarge or shrink these windows and switch between them at will. Saving groups of functionally related windows together as a workspace would help preserve the context of an activity in units which are more meaningful to users.

Maintaining records of activities

The history of a user's activity within a workspace constitutes a record of the command sequence for performing that activity. We call this command sequence an *Activity Script*. Activity scripts are useful whenever a user wants to perform an activity similar to something done before. If the desired activity is exactly the same as a previous one, the user should be able to redo the old activity script. If there are only small differences, the user can edit the script and then execute it. If there are major differences, the old script can still be very useful as a guide to the new activity. No special effort should be involved in creating an activity script, for they are simply records of transactions. Activity scripts could be used to support sophisticated redo/undo facilities which would act on sequences of commands at the task level, rather than just on single command lines.

Functional groupings of activities

We expect that users will want to organize their workspaces to reflect functional groupings. All material associated with a given activity or large group of activities could then be accessed as a unit allowing more continuity between work sessions. For example, the functional relationship between elements of a workspace may be as loosely defined as the group of tasks the user desires to accomplish over the next few sessions. The workspace contents, in this case, function much like a "To Do" list. The potential for hierarchical grouping of workspaces means that this "To Do" workspace might have any number of lower-level workspaces within it that could be maintained in any state of completion, perhaps with notes as reminders of what to do next. When the workspace is reactivated, the user would be shown where work was discontinued, and the activity history list would be available for reference.

Multiple perspectives on the work environment

One consequence of creating complex work environments is that users often lose track of their goals and current location within the system. Multiple window displays do not in themselves solve this problem, as the "messy desk" phenomenon can appear with a vengeance. We address this problem by proposing a set of tools—and ultimately displays—that allow users to organize their activities. The rich interconnections among activities require a support system that allows users to have multiple perspectives on their activities. Such perspectives would include displays indicating activity sequences based on such measures as temporal ordering and goal ordering, this latter being useful in keeping track of the many subtasks necessary to achieve a goal. Possibly other information (eg., the temporal dependencies between tasks) could also be presented. [2]

2. The Apple LISA computer system, for instance, has a PERT program that displays the temporal dependencies existing between different tasks.

Interdependencies among items in different workspaces

Within a complex goal-structured environment, many items (eg. text files and activity scripts) may be important for more than one task. In practice, this means that very often the user may want to assign a portion of one workspace to another workspace. It is therefore important to support the ability to have multiple instances of a particular file or note.

Updating one instance of such an item, however, leads to the question of whether or not to generalize this update to all or several instances. In any viable system the user should be able to trace the relations between workspaces and files and be able to locate multiple instances of a particular item. This facility might enable the system to query the user whenever an item duplicated in other workspaces is updated.

Summary

In our empirical data, the annotations that users added to their history lists showed that they view their interactions with the computer in terms of goals rather than system commands. Accomplishing these goals involves translating them into a series of commands, actually executing the commands, and evaluating the result for success. In the course of this process, users often become confused about their overall goals or lose track of their secondary goals. Our plan is to provide users with a system for managing activities which maps well onto the users' own goal structures, thus reducing the mental load on the users.

Our current research activities focus on both empirical and theoretical developments. We have built a working system (Notepad) to explore in detail the issues involved in how best to postpone certain housekeeping chores while the user is busy creating new text. [3] This system also allows us to explore questions concerning how to manipulate, save and retrieve contexts. In addition, we are developing the concept of a workspace as an integrating idea that has implications for human-computer interface design.

---

3. It is useful here to introduce another aspect of the work on activity structures. Cypher has designed a system called Notepad for collecting and organizing textual notes. The principles of organization embodied in Notepad are central in complex activity structuring. A note can be input into the system without first specifying a name or the location of where it should be stored. The note appears as a subset of the current environment and is identified by a number (temporal information). If the note is left uncompleted when the user switches to another task, the system signals the user that there is an uncompleted task and supports reactivating the note for further manipulation. At any time, the note may be named, renamed, or relocated. Notes are hierarchically organized with mechanisms which allow the user to specify the structural family in terms of parents, children, and sibling notes. A note can have multiple parents, meaning that it can be available within any appropriate context and can be easily retrieved.

References

Huff, K. E., & Lesser, V. R. Knowledge-Based Command Understanding: An Example for the Software Development Environment. Amherst, Massachusetts: Computer and Information Science, University of Massachusetts, Amherst. June 30, 1982. (Technical Report 82-6.)

Perlman, G. Two Papers in Cognitive Engineering: The design of an interface to a programming system, and MENUNIX: A menu-based interface to UNIX (User manual). La Jolla, California: Center for Human Information Processing, University of California, San Diego. November, 1981. (Report No. 8105.)

Smith, D. C., Irby, C., Kimball, R., & Verplank, B. Designing the Star User Interface. *Byte*, 1982, *7* (No. 4: April), 242 - 282.

Teitelman, W., & Masinter, L., The Interlisp programming environment. *Computer*, 1981, *14*, (April, No. 4), 25 - 33.