Automatic Chunk Detection in Human-Computer Interaction

by

Paulo J. Santos
and
Albert N. Badre

GIT-GVU-94-4
January 1994

Graphics, Visualization & Usability Center

Georgia Institute of Technology
Atlanta GA 30332-0280

AUTOMATIC CHUNK DETECTION IN HUMAN–COMPUTER INTERACTION

Paulo J. Santos , Albert N. Badre

Graphics, Visualization, and Usability Center
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
United States of America

E-mail: {pas,badre}@cc.gatech.edu

ABSTRACT

This paper describes an algorithm to detect user's mental chunks by analysis of pause lengths in goal-directed human-computer interaction. Identifying and characterizing users' chunks can help in gauging the users' level of expertise. The algorithm described in this paper works with information collected by an automatic logging mechanism. Therefore, it is applicable to situations in which no human intervention is required to perform the analysis, such as adaptive interfaces. An empirical study was conducted to validate the algorithm, showing that mental chunks and their characteristics can indeed be inferred from an analysis of human-computer interaction logs. Users performing a variety of goal-directed tasks were monitored. Using an automated logging tool, every command invoked, every operation performed with the input devices, as well as all system responses were recorded. Analysis of the interaction logs was performed by a program that implements a chunk detection algorithm that looks at command sequences and timings. The results support the hypothesis that a significant number of user mental chunks can be detected by our algorithm.

KEYWORDS: human-computer interaction, novice/expert differences, chunking, chunk detection, models of the user, event logging, user study.

INTRODUCTION

Users of computers vary widely in knowledge, skill, and abilities. Additionally, any individual user's skill and abilities vary over time, as users learn through practice and experience. User diversity and variability have been the object of many studies, classifying users into categories corresponding to levels of expertise. Through our work, we show how an automatic identification of computer user skill can be achieved through chunk identification.

In the human-computer interaction literature, users are typically classified as novices or experts. We prefer not to stereotype users into two hermetic categories, but to classify them based on their abilities to perform specific tasks. A user may be an expert at some task, yet be a novice at another task, even if the two are closely related. More important than labelling users as "novices" or "experts" is to determine what users know about an interface, the tasks they perform, and how they combine their knowledge whren interacting with the computer system.

User behaviors are intimately related with their cognitive skills. According to Anderson's ACT* theory [1, 2], humans involved in goal-directed problem solving situations typically perform according to one of three skill levels: interpretive, knowledge compilation, or procedural. In the interpretive stage, people apply domain-independent knowledge to domain-specific declarative information to derive a procedure that will be suitable to solve a problem, using weak methods of problem solving. The knowledge about the skill is used interpretatively using only declarative information. This interpretative problem solving mechanism is often effective in solving problems, but requires a significant amount of resources, both in terms of time and working memory demands. Through repetition of the interpretative procedure, knowledge compilation occurs, effecting the transition to the procedural stage. In the procedural stage, specialized production rules enable the performance of the skill faster (due to faster access time for retrieval and to the tight bindings of actions) and with lower demands on working memory (by retrieving just one procedure without declarative details, which were lost through proceduralization). As an example, a positive effect of compilation that leads to improvement in skill performance can be found in a famous Shiffrin and Schneider experiment [22], where set size effects are seen to disappear in the scan task. After compilation, people start performing the task according to the procedural stage. Task execution is faster, as retrieval time is shorter and the entire sequence of steps has been precompiled. During the procedural stage, execution time and error rate may continue to decrease wiith practice [19, 3].

Of the three cognitive stages considered in ACT* [1, 2], knowledge compilation is one of short duration that simply enables the transition from the declarative to the procedural stage. The declarative and procedural stages have a permanent character, and their performance can be observed repeatedly.

A body of literature, generally referred to as the chunking literature, establishes relationships between cognitive stages and observable behavior. A chunk is a coherent cognitive unit that the user holds in short term memory while performing a cognitive operation, and would roughly correspond to one production in ACT*. Results such as those found in [12, 23, 5, 4, 7] show that experts organize the components of their mental process in larger chunks than novices. The chunk boundaries can be identified by pauses in the observed behavior. The feasibility of chunk identification by pauses has been demonstrated in several domains, including games [12, 21] and tactical decision scenarios [5].

In our work, we apply chunk detection techniques to human-computer interaction. We establish a relationship between the observable user behavior and the assessment of the user's chunks by recording interaction, analyzing event contents and pauses between events, then inferring user chunks.

DETECTING CHUNKS IN INTERACTION

When interacting with a computer, users typically behave according to the acquisition/execution cycle [111. During the acquisition phase, users think of a goal, formulate a strategy to reach their goal, and plan the execution of the strategy on the computer. Then they proceed to the execution phase, in which they carry out the plan by interacting with the computer. Once they have reached their goal, the cycle repeats itself with a new acquisition phase. During execution, the execution plan is stored in the user's short term memory, and is referred to as a chunk.

In the acquisition phase, people are immersed in a problem solving situation, and there is typically no physical interaction with the computer. Significant research efforts have been devoted to studying how people solve problems, and how they learn problem solving. In fact, Anderson asserts that the study of problem solving and learning have occupied American psychology for much of this century [3]. In our work, we do not attempt to analyze problem solving and learning. We rely on the fact that people plan an execution strategy during acquisition, and that such planning takes time. During that time, they are generally not physically interacting with the computer.

During execution, a burst of activity occurs. If one looks at the distribution of the activity in an interaction log, one will find that generally pauses correspond to the user's acquisition phases and activity corresponds to the execution phases. The contents of the bursts of interaction delimited by pauses are chunks.

Some models have been proposed for the estimation of the execution time of tasks by experts, e.g. [10, 16]. These predictive models are useful for predicting expert performance because expert chunks are large. They are less powerful in modeling novices because the chunks formed by novices are much smaller, typically resulting in trivial execution plans that even the simplest model could predict.

In order to identify user chunks, we will use the results of predictive models for execution time. In particular, we start by using a variation of the Keystroke-Level Model [11] as a first approximation. Other models may be added later using a similar approach.

The general principle of the chunk detection algorithm is as follows:

- Assume expert behavior.
- For each sequence of events in the interaction log, apply the predictive model to predict execution time.
- If a pause in the interaction log cannot be justified by the predictive model, assume that the pause was caused by a shift of the user to the acquisition phase, and assess a chunk boundary at that point.

We realize that there may be overlapping between processes. Users often engage in multiple concurrent processes, spending some of their time in each process [13]. One consequence of the coexistence of multiple processes is that the user may not need to wait for the conclusion of a process to start thinking about the next one. in such a situation, the tail end of an execution may overlap with the beginning of the next acquisition phase. This overlap may reduce the length of the pauses delimiting the chunks. We hypothize that the overlap is small compared to the typical length of the acquisition phase. Furthermore, it seems reasonable to assume that the time spend switching between activities should be much shorter that the time spend in each activity. If our assumptions hold, we will still be able to identify chunks based on the pauses that surround them. The importance of our algorithm for chunk detection derives from the fact that it works based exclusively on information that is available to the computer. It does not require external human analysis.

To limit the scope of our work, we consider only interaction with a classical computer, with a keyboard and a mouse as input devices, and a raster monitor as the output device. This limitation is not severe, as it considers a most computers in use for information processing today.

For such a system, the time for task execution will be based on a variation of the Keystroke-Level Model. The predicted expert performance t is given by

$$t = t_K + t_P + t_H + t_R + t_S \quad (1)$$

The term t_K represents the time to press a key or mouse button. That time depends, among other factors, on the users typing skills, familiarity with the keyboard, and physical characteristics of devices. Physical characteristics of the keyboard and mouse can be obtained as part of the system information. User typing skills vary slowly over time according to the power law of practice [19] with a low value of α, the learning rate. The flat slope of the practice curve affords the estimation of the user's typing speed by analyzing recent samples of typing by that user on the system. t_K is inversely proportional to the estimated user typing speed.

The term t_P represents the time to move the pointer from the current position to the target position. This time is approximated by Fitts's law [15], which states that the time, in seconds, to move the hand a distance d to a target of size s is given by:

$$t_P = 0.8 + 0.1 \log_2 (d/s + 1/2) \quad (2)$$

Given that the values 0.8 and 0.1 in equation 2 are approximations found by Fitts in his experiments, and that these values vary with system characteristics and user skills, we prefer to use a parametrized variation of Fitts's law.

$$t_P = f_0 + f_1 \log_2 (d/s + 1/2) \quad (3)$$

The parameters f_0 and f_1, as with the typing speed parameter, vary slowly over time. Like the typing speed, they can also be estimated. We use logarithmic regression analysis to estimate the values for f_0 and f_1 based on recent interaction by the user.

The term t_H in equation 1 accounts for time to switch devices. In typical desktop systems of today, this will be the time to move the hand from mouse to keyboard or vice-versa. Therefore, t_H is a function of source and target devices, expressed as $t_H(\partial_s,\partial_t)$. Although this seems to contradict Fitts's law, experiments have obtained values in the order of 400 milliseconds to move the hand from keyboard to mouse [9]. This value may be a good approximation for $t_H(\mathtt{k},\mathtt{m})$ and $t_H(\mathtt{m},\mathtt{k})$ for experienced users.

The term t_R represents system response time. On the assumption that all the activity in the system is controlled by the user, all the output by the computer is as a response to a previous user action. However, there is no guarantee that a system output is a response to the last user action, as the user may be "typing ahead" or performing operations disregarding system output. In any case, we can say that when system output occurs, the system response time is at least the difference between the time that the output occurred and the time of the last user input event.

The term t_S represents time for the user to locate information on the screen. All the information presented has to be expected, yet the user may still need to find its location. An example is a confirmation dialog box. Let's suppose that a user wants to delete a file, and that the system requires confirmation of the file deletion. The user may be expecting the confirmation dialog box to appear after the delete command, yet he or she may not know exactly where the box will appear, or where the OK button will be located inside that box. However, the user had already formulated the plan to press the OK button and is fully expecting it. t_S accounts for the scan time for the user to find the OK button. Scan time depends on essentially three factors: the amount of information presented to the user, its dispersion on the screen [20], and its visual properties. Scan time should increase with the amount of information and with dispersion. Visual properties, such as size and color of the objects, also affect scan time.

A direct way to measure scan time is through the use of eye-tracking devices. With these devices, the system could measure scan time, and account for t_S in equation 1. However, eye-tracking devices are still not easily accessible to most interface developers, let alone end users. In anticipation of a probable wider availability of eye-tracking devices, we have included scan time in our chunking model.

Our chunk detection algorithm takes an interaction log and produces a set of chunks based on the events. A chunk is completely defined by a pair of event timestamps corresponding to the beginning and end events of the execution phase of the chunk. With the event log and the pair of timestamps, one can obtain the contents of the chunk.

The chunking algorithm works as follows. For each contiguous pair of events E_i and E_{i+1} with timestamps T_i and T_{i+1}. evaluate the following logical expression:

$$T_{i+1} - T_i > t(E_i,E_{i+1}) + \Delta \quad (4)$$

Δ is a positive number representing a tolerance factor of the chunking algorithm, which accounts for imprecision in the timing and small fluctuations in the parameters. Basically, it indicates how long a user is allowed to pause beyond the normal execution time and still have the two events be considered part of the same chunk. It can be a constant, or a function of previous chunk behavior. For example, if, by chunk analysis of previous interaction, the chunking system had detected that the user had consistently mastered a certain operation and usually processes it as one chunk, then it could increase the value of Δ to allow for some additional tolerance. The value $t(E_i,E_{i+1}) + \Delta$ become a threshold against which inter-event pauses are evaluated.

Let us illustrate the calculation of $t(E_i,E_{i+1})$. Consider the following two consecutive events in the interaction log:

```
Time     EventType       Key    Button    PointerLoc    Object
22800    KeyRelease      P                311,208       Window:Untitled
22850    ScreenChange
24460    MousePress             1         203,40        Button:Center
```

This example shows two user events (key release and mouse press), with timestamps of 22800 and 24460 respectively. Between the two events there is a screen change, which is probably a system response to the user's pressing of a key.

$t(E_i,E_{i+1})$ is given by the sum of the terms shown in equation 1: $t_K + t_P + t_H + t_R + t_S$. K is estimated by the average time to press the mouse button (let's say, 400 milliseconds). t_P is the time to move the pointer from its previous position to its current position. Considering that the size of the target (the button "Center") is 16 pixels, and that f_0 and f_1 have the standard values of Fitts' law (800 and 100 ms respectively), equation 3 yields a value of 1170 ms for t_P. t_H takes of the value of t_H(k,m), which is 400 ms. t_R, the system's response time to the previous user action, is the time it takes for the system to provide feedback to the user's pressing of P. In this case, the value of t_R is the difference in the timestamps of the key press event and the screen change, or 50 ms. Finally, we consider a scan time t_S of zero. Adding the terms together, we obtain a value for $t(E_i,E_{i+1})$ of 400+1170+400+50+0, or 2020 ms.

In this case, since $T_{i+1} - T_i$ (1660 ms) is less than $t(E_i,E_{i+1})$, the algorithm will keep the two event in the same chunk, regardless of the value chosen for Δ.

In the current version of the algorithm, we are using a constant for Δ. Although its value is modifiable, we have been using the value of 400 ms. As we gather information about the user's knowledge, as well as experience with the algorithm, that information may be used to determine the probability of a chunk boundary. The chunking algorithm would then become a probabilistic one, providing answers to "given the time between events, the task at hand, and the user's previous performance and this task, how likely is it to have a chunk boundary between E_i and E_{i+1}?" In the probabilistic algorithm, Δ may not be a constant but a function of context and user knowledge.

EMPIRICAL VALIDATION

We designed an experiment to verify the validity of our chunk detection algorithm. The purpose of this experiment was to show that mental chunks can indeed be inferred from an analysis of human-computer interaction logs.

This study is necessary because we have not found empirical evidence to the detectability of inter-chunk pauses in goal-directed human-computer interaction. It is possible that extraneous effects such as those introduced by characteristics of physical devices (such as a specific keyboard design) or of the tasks themselves (such as interruption in multitasking) outweigh the inter-chunk pauses. In that case, the inter-chunk pauses would be buried in "noise", and would be difficult to detect. Thus, in this experiment we test whether chunks can be detected by an analysis of interaction, in particular by an analysis of pauses.

Method

In this experiment, we monitored users performing a variety of ill-defined goal-oriented tasks using several computer applications and interaction devices. We record their actions using CHIME [6], logging every command invoked and every operation performed with the input devices, as well as system responses. We also record the entire interactive session on video for independent analysis.

From a pool of sixty undergraduate students in the Georgia Institute of Technology, twenty-eight were used as subjects for this experiment, receiving class credit. They performed a task for a maximum of 15 minutes on an Apple Macintosh computer.

Subjects were divided into three groups, based on task. Group T1 composed a text using Microsoft Word; group T2 used Microsoft Word to format an existing unformatted text; and group T3 used MacDraw to decorate the interior of an apartment. All three tasks were ill-defined and open-ended. Subjects were not given detailed specifications of the tasks, and were asked to perform them until they felt they were satisfied with the results (with a 15-minute maximum time limit). Task T1 essentially explored use of the keyboard in a text-processing environment, allowing for a comparison to independent research results, including work by Card, Moran, and Newell [10] and Bovair, Kieras, and Polson [8]. Task T2 explored use of both keyboard and mouse, with emphasis on direct manipulation of interface objects, such as selection of text, and use of menus, dialog boxes and icons. Finally, task T3 explored the use of a direct manipulation graphic package, with high emphasis on mouse operations. The design of the experiment considers a between-subject distribution of the tasks, avoiding effects of learning and transfer.

The tasks were chosen as three simple tasks that exercise a significant portion of the operations performed in typical interactive environments. Task T1 explores basically the the use of the keyboard for typing. Task T2 is rich in the use of traditional interactors such as pull-down menus, scroll bars, dialog boxes, buttons, and selection of objects by pointing and draging. Finally, task T3 is rich in direct manipulation of graphical objects, including operations of creation, positioning, and resizing of graphical objects. Our intent in choosing these tasks was to provide a rich coverage of typical interactive operations.

Our goal is to show that our chunking algorithm performs well in all situations, for all three tasks.

In addition to the interaction, we attempt to know why subjects do the actions they do and how they form their chunks. We used verbal protocols to gather that information. Verbal protocols were collected both during the interaction ("think aloud“) and immediately after ("retrospective") the interactive session.

Verbal protocols do not tell us what users think, they just tell us what users say [14]. A good match between what users think and what they say is obtained if we ask them to "think out loud". Therefore, in this experiment we record think aloud protocols. However, there are two potential problems with "think aloud" protocols: first, the verbalization may change the way in which people think, as it may help or hinder in their organization of ideas; and second, the time required for the verbalization may have an impact on the pauses, creating an undesired source of noise in our data about inter-chunk pauses. Considering that timestamps are accurate to within 17 milliseconds, it is unclear to what extent verbalization may affect the accuracy of the results derived from the timestamped data. In this experiment, we measure the effects of "think aloud" protocols in the data collected by monitoring human-computer interaction. Verbalization is a carefully controlled variable in this experiment.

In this experiment, we control for effects of verbal protocol with a two-level between-subjects design. One group, VDA, verbalized during and after the interactive session. The other group, VA, verbalizes only after the session. The interactive session data of the VA group in uncorrupted by the verbalization, but the quality of the verbalization may not be as good as that of the VDA group.

To summarize, the experiment considers two independent variables: verbalization phase and task. Figure 1 illustrates the experiment design.

		Verbalization	
		After	During & after
Task	Task 1: text entry		
	Task 2: text formatting		
	Task 3: drawing		

Figure 1 — Experiment design

Procedure

After the introductions and consent form, subjects were given instructions about the experimental session, which were read to the subject by the experimenter. Subjects were asked brief questions to assess their familiarity with the equipment and the software. Those with no previous experience using the Macintosh or the mouse were given a brief explanation of its operation; those with no previous experience using the application were also given brief instructions about its operation.

Subjects were then told about the task they were expected to perform. The scope of the task was ill-defined in order to allow them to form their own goals and strategies. One of our goals was to try to simulate realistic work conditions, and user-defined goals are predominant in human-computer interaction. The task domains of all the three tasks were such that all subjects were familiar with them, although not necessarily experts. T1 involved writing about their experiences as students in one typical academic quarter at Georgia Tech; in T2, they were asked to beautify the formatting of a one-page class syllabus, by formatting it; and task T3 required them to decorate a small apartment, given its floor plan, by drawing representations of pieces of furniture or other decorative items on top of the floor plan.

Subjects were then given opportunity to ask questions. Clarification questions about the experiment or the task objectives were answered. Questions about the operation of the equipment or the software were not answered, other then repeating what was in the instructions.

They were told that they could do as little or as much as they wanted, but that they should plan to finish within fifteen minutes. Those who did not finished by the 15-minute deadline were interrupted. Subjects in group VDA were told to think out loud, as if trying to explain what they were doing to a work companion.

During their interaction with the computer, CHIME [6] recorded all their operations with the interaction devices, as well as all the command invocations and operations on objects. Every operation was time stamped with the maximum clock resolution of the Macintosh (approximately 17 milliseconds). Audio and video recordings of the sessions were also obtained, with the camera focusing on the screen. The experimenter stayed in the room during the interactive session, but did not interfere with the subject's operation except to remind subjects to think out loud when they seemed to forget.

Immediately following the interactive session with the computer, the subjects were shown the video playback of what they had done. Subjects sat approximately 10 feet away from a 27-inch screen. Sound was muted, so that subjects in the VDA group could not hear what they had said. Subjects were asked to comment what they were thinking about as they were interacting. Their comments were recorded synchronously with the video playback.

Results

In encoding the verbal protocols, goals expressed by the subjects were related to their corresponding begin and end events in the interaction log. This encoding was done based on the events and not on their timestamps. For example, the subject might have said "Now I am going to draw a table," and then proceeded to select the rectangle tool, and draw a rectangle. In this case, we would have encoded the event corresponding to the mouse press on the rectangle tool area as the beginning of the chunk, and the event corresponding to the release after the rectangle drag as the end of the chunk.

In the analysis of this experiment, we seek to find answers to two questions:

- are mental chunks detectable exclusively by analysis of the recorded interaction log?
- how does "think aloud" verbalization affect performance in human-computer interaction, and particularly the formation of chunks and the duration of inter-chunk pauses?

The answer to the last question can be found by comparing the interaction of VDA and VA subjects. Differences between VDA and VA data provide information about how the verbalization during the session affects their task performance and mental processes.

We tested the effect of the verbalization phase on a number of dependent variables: number of user-generated events, number and size of chunks in protocol, and length and duration of chunks. The number of user-generated events ins the total count of key and mouse presses and releases in session. The number of chunks in the protocol is the count of the chunks that were identified fro the subjects' utterances. The size of the chunks in the protocol is the number of user-generated events that correspond to the execution of the chunks describe by the subjects' utterances. We could not find significant effect of verbalization on any of these dependent variables, except in the total number of events. As expected, there were fewer events in think aloud than in retrospective conditions (average 530 versus 883, $F_{2,22}=36.83$, $p<0.01$).

The fact that there was no significant effect of verbalization in the characteristics of the chunks between the two verbalization conditions is important. It tells us that we can proceed with the analysis of both VD and VDA conditions in the same manner, and that we do not have to analyze different in "during" and "after" protocol in the VDA condition.

The answer to the first question, "Are mental chunks detectable exclusively by analysis of the monitoring log?", is the fulcrum of this experiment. The analysis of the data was done by comparing the chunks identified in two ways, from two sets of data: by analysis of command sequences and their timings in the interaction log, and by analysis of the verbal protocols.

Analysis of the interaction logs is performed by a program that implements the chunk detection algorithm presented earlier. The algorithm looks at command sequences and timings. When a pause between two operations is greater than can be accounted for by model, it assumes that the user is engaged in a cognitive operation that has not been previously compiled, and assesses a chunk boundary.

The algorithm uses a parameter for the typical time between keystrokes (t_K) and two parameters (f_0 and f_1) for pointer movement estimation. These parameters were estimated for all subjects. Considering the data for the 15 subjects that typed any significant amount, t_K averaged 339 ms, with a standard deviation of 105 ms. The values correspond to typing speeds between 23 and 55 words per minute, which are reasonable numbers for our population. The estimated values of f_0 for the 21 subjects that use the mouse significantly varied between 507 and 1039 ms, with a mean of 765 ms and a standard deviation of 143 ms. For those subjects, f_1 varied between 134 and 325 ms, with mean 229 ms and standard deviation of 46 ms. f_0 is very much in line with the value of 800 in Fitts' law [15]. However, f_1 is significantly higher than Fitts' value of 100 ms. This difference may be due to physical properties of the mouse or the surface on which it moved.

Along with the parameters t_K, f_0 and f_1, which were estimated for each user as explained above, we set the constants t_H (homing time) to 400 ms [9], t_S (scan time) to zero, and Δ (tolerance) to 400 ms. The chunk detection algorithm was then applied to the logs of every session, and the resulting chunks were recorded.

In parallel, we encoded the verbal protocols, and map the tasks that the subjects verbalized to chunks. Many significant research results on chunking and skill acquisition are based on experiments that rely on results of verbal protocols, including [12, 21]. Lacking a better approach to extract images of the knowledge from the subject's mind, we analyzed the verbal protocols and relied on them for chunk identification.

In encoding the subjects' utterances into chunks, the encoder had to make inferences when interpreting of the utterances. Occasionally, the utterances led to two or more possible interpretations. In those cases, to avoid results that would biased by the subjective interpretation, the chunk was discarded (see the end of the next paragraph for a discussion of discarded chunks.) It may further be argued that, even when only one interpretation of an utterance is reasonable, the utterance may correspond to one explicit and one or more implicit chunks. Often, the implicit chunks would probably correspond to a decomposition of the overall chunk that the subject was describing in the explicit chunk. We assume that, if subjects had been able to reason far enough to combine the various implicit chunks into one larger explicit chunk, then they could just as well have formulated the execution plan for the larger explicit chunk. Therefore, we encoded the utterances only into their explicit chunks.

To compare the chunks identified from the protocol with the chunks detected by the program, we devised a scoring system, which is illustrated in Figure 2. The horizontal bars represent chunks. Case a) illustrates a full match: the algorithm detected exactly the chunk that was identified in the protocol. We assign a score of 1 to this match. Cases c) through f) represent partial matches: the algorithm detected either the beginning or the end of the chunk. Since a chunk consists of two boundaries, it seems reasonable to assign half of the full score to each end. Therefore, we assign a score of 0.5 to these matches. Case b) is intermediate between a full match and a partial match. In this case, the algorithm matches both beginning and end of a chunk, but introduces one or more additional breaks. Given that this situation is better than c) through f) but worse than a), we assign it a score of 0.75. Cases g) and h) correspond to mismatches. In case g), neither the beginning nor end of a chunk are identified by the algorithm, and so a score of 0 is appropriate. Finally, case h) corresponds to interaction that was recorded but for which there is no protocol chunk. This case occurred about 75% of the time, because users do not verbalize every action that they do. Since there is no way to know whether the algorithm is right or wrong, we discarded all instances of case h). Discarding such a large percentage of the data is not a problem, in our case. This is because we verified that there was no significant difference in the subjects' behavior between the VD and VDA conditions. Therefore, as far as the ability to detect chunks is concerned, there is no effect of verbalizaton. If this is the case, discarding data that depends on the verbalization does not affect the analysis of the overall resuts. In this case, the only potential problem with discarding data is a larger experimental error (due to fewer data points), but this problem can be overcome simply by running more replicates of the experiment.

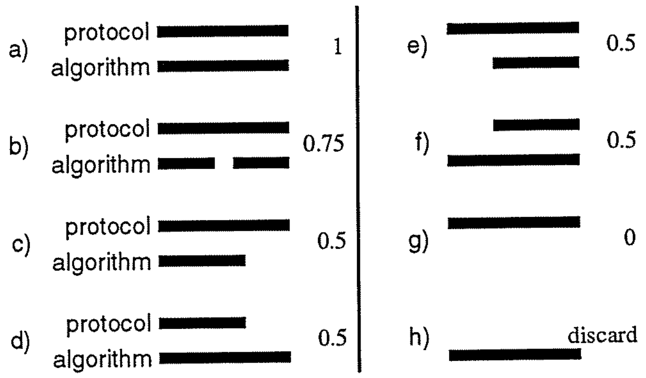

Figure 2 — Scoring the matching between the chunks identified in the protocol and detected by the algorithm

In the combined 28 experimental sessions, 507 chunks were identified from the protocol. Of those 507 chunks, 201 were full matches (case a), 126 were full matches with breaks (case b), 146 were partial matches (cases c through f) and 34 were not matched at all (case g). The overall score was 0.74.

Although we can not find statistical significance of any effects of verbal protocol in the chunking performance, we detect some trends. As expected, the chunk detection algorithm performed better with think aloud protocol (average 0.76) than with retrospective protocol (average 0.71). Note that the difference is not significant.

There was a significant effect of task ($F_{2,22}$=4.37, $p<0.05$) on the score of the matches. The algorithm performed significantly better on the typing task (average score 0.84) than on the formatting (0.74) or drawing tasks (0.66). However, if we consider only the completely matched chunks, we can not detect any significant effect of task ($F_{2,22}$=1.10).

DISCUSSION

The results of this experiment are encouraging, and the overall 73% score supports our hypothesis that a significant number of user mental chunks can be detected by our algorithm. It is especially significant that, of the 73% total score, over 54% corresponds to chunks fully matched by the algorithm. The use of a variable pause duration threshold given by equation 4 was powerful. For comparison purposes, using a separate simple algorithm, we tested several fixed duration times and used those as threshold for assessing chunk boundaries. For the data we collected, we found that the best value for a fixed threshold was at 2.15 seconds. In that best case, the fixed threshold algorithm was able to identify correctly only 96 (19%) of the chunks. That is less than half the performance of our algorithm with a variable threshold.

The fact that there were few differences due to protocol was surprising. We expected additional or longer pauses in the think aloud condition, which would contribute to a better chunk detection. Fortunately, the data does not confirm this hypothesis, as there is no significant difference in the pause duration distributions of the two conditions. We still believe that the information about user goals is more truthful in think aloud protocols than in retrospective, because subjects giving retrospective protocols may have forgotten what their exact goals had been during interaction.

The significant effect of task on chunk detection performance can be explained by several factors. In the drawing case (task T3) it was often hard to exactly determine the beginning and end event. When the subject says "I am going to draw a table," he or she may consider the task completed when the table is placed, or maybe only after the handles have been dismissed by clicking outside any object. We do not know exactly what the subject means by "drawing a table," and we may make assumptions that do not match those of the subject. In task T1 there were fewer such cases of possible ambiguities in interpretation, because most of their protocol was sentence oriented, and corresponded to sentences in their text. Tasks T2 and T3 suffer from yet another problem: the tasks to be performed are not language related, as in task T1. Therefore, it is reasonable to expect a greater separation between the protocols and the actual intentions of the subjects. Finally, there is task size. In task T1, subjects typically built chunks of one of more independent clauses at a time. Typing independent clauses takes some time, and that time is sufficient for them to express their verbalization. However, in tasks T2 and T3 some actions were very fast: selecting a piece of text, a graphical object, or pulling down a menu. Subjects may have been tempted to summarize their thoughts in order to make the verbalization fit into the execution time. For example, a user may structure the activity of drawing a table as three separate chunks: selecting the rectangle tool, determining a location to place the table, and finally placing it. However, they will often just say that they are drawing the table. This results in a full match with breaks (score 0.75), although the algorithm was probably right and the verbalization was probably wrong.

CONCLUSIONS

The automatic detection of the user's mental chunks can be used for several types of applications. The first obvious application is in adaptive interfaces. In most cases, the goal of interface adaptation is to better match the user's knowledge or the tasks to be performed. The first step in any system-initiated form of adaptation, including total self-adaptation, is system initiative [17]. The system has to perceive a need for adaptation in order to initiate it. The next two states of interface adaptation considered by Kühme et al [17] are proposal and decision. In the decision stage, alternatives for adaptation developed in the proposal stage are evaluated against an adaptation strategy and knowledge about the user, the application, and the constraints. In all three stages, knowledge about the user is essential for adaptation.

It is possible, although this would need to be tested, that pause durations are an indicator of the user's next activity. The different complexities of the different cognitive processes leading to the execution of a task require different acquisition times by the user. If the number of possible tasks is limited, the acquisition times for each can be predicted (with models based on such theories of cognition as Soar [18]). Once a chunk boundary has been assessed by a chunk detection algorithm, the system could anticipate the next action based on the time that the user is taking to perform it.

Chunk detection technique is also useful for long-term studies of learnability. As users become more experts, their chunks become larger, longer, and more regular [5]. A study of user chunk evolution using an application over an extended period of time should be a good indicator of the ease of learning of its interface. Such studies are not typically performed today due to the high costs and manpower requirements associated with them. Using automated interaction logging, such as provided by CHIME, and automatic chunk detection, the entire process is automated, and becomes affordable.

ACKNOWLEDGEMENTS

This research was partially supported by a grant from Intel.

REFERENCES

[1] Anderson, J.R., Acquisition of cognitive skill. *Psychological Review 89*, 4 (1982), 369-406.

[2] Anderson, J.R. *The architecture of cognition,* Harvard University Press, Cambridge, Massachusetts , Vol. 5, Cognitive Science Series(1983).

[3] Anderson, J.R., Problem solving and learning, *American Psychologist 48*, 1 (Jan 1993), 35–44.

[4] Badre, A.N., Designing chunks for sequentially displayed information. In Badre, A. and Shneiderman, B. (Eds.), *Directions in Human/Computer Interaction.*, Ablex, pp. 179-193, Norwood, New Jersey, 1982.

[5] Badre, A.N., Selecting and representing information structures for visual presentation. *IEEE Transactions on Systems, Man, and Cybernetics 12*, 4 (Jul/Aug 1982), 495-504.

[6] Badre, A.N. and Santos, P.J., CHIME: a knowledge-based computer-human interaction monitoring engine, Technical report no. GIT-GVU-91-06, Atlanta, Georgia, Jul 1991.

[7] Barfield, W., Expert-novice differences for software: implications for problem-solving and knowledge acquisition. *Behaviour and Information Technology 5*, 1 (1986), 15-29.

[8] Bovair, S., Kieras, D.E., and Polson, P.G., The acquisition and performance of text-editing skill: a cognitive complexity analysis. *Human Computer Interaction 5*(1990), 1-48.

[9] Card, S.K., English, W.K., and Burr, B., Evaluation of mouse, rate controlled isometric joystick, step keys, and text keys for text selection on a CRT. *Ergonomics 21*(1978), 601-613.

[10] Card, S.K., Moran, T.P., and Newell, A., Computer text-editing: an information-processing analysis of a routine cognitive skill. *Cognitive Psychology 12*(1980), 32-74.

[11] Card, S.K., Moran, T.P., and Newell, A., The Keystroke-Level Model for user performance time with interactive systems. *Communications of the ACM 23*, 7 (Jul 1980), 396-410.

[12] Chase, W.G. and Simon, H.A., Perception in chess. *Cognitive Psychology 4*(1973), 55-81.

[13] Cypher, A., The structure of users' activities. In Norman, D. and Draper, S. (Eds.), *User Centered System Design: New Perspectives on Human–Computer Interaction*, Lawrence Erlbaum Associates, pp. 243–263, Hillsdale, New Jersey, 1986.

[14] Ericsson, K.A. and Simon, H.A. *Protocol analysis: Verbal reports as data,* MIT Press, Cambridge, Massachusetts , Bradford Books(1984).

[15] Fitts, P.M., Perceptual motor skill learning. In Melton, A.W. (Ed.), *Categories of human learning*. Academic Press, pp. 243-285, New York, 1964.

[16] Kieras, D.E. and Polson, P.G., An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies 22*, 4 (Apr 1985), 365-394.

[17] Kühme, T., Dietrich, H., Malinowski, U., and Schneider-Hufschmidt, M., Approaches to adaptivity in user interface technology. In *Proceedings of the IFIP WG2.7 Working Conference on Engineering for Human-Computer Interaction,* Unger, C. and Larson, J.A., IFIP, Elsevier, Amsterdam, August 1992.

[18] Laird, J.E., Newell, A., and Rosenbloom, P.S., SOAR: An architecture for general intelligence. *Artificial Intelligence 33*, 1 (1987), 1-64.

[19] Newell, A. and Rosenbloom, P.S., Mechanisms of skill acquisition and the law of practice. In Anderaon, J.R. (Ed.), *Cognitive skills and their acquisition.* Lawrence Erlbaum Associates, Ch. 1, pp. 1-55, Hillsdale, New Jersey, 1981.

[20] Olsen, Jr., D.R., A programming language basis for user interface management. In *Proceedings of the CHI '88 Conference on Human Factors in Computing Systems,* ACM/SIGCHI, ACM Press, Washington, D.C., May 1988, pp. 171-176.

[21] Reitman, J.S., Skilled perception in Go: Deducing memory structures from inter-response times. *Cognitive Psychology 8*(1976), 336-356.

[22] Shiffrin, R.M. and Schneider, W., Contreolled and automatic human information processing: II. Perceptual learning, automatic attending, and a general theory. *Psychological Review 84*(1977), 127-190.

[23] Shneiderman, B., Exploratory experiments in programmer behavior. *International Journal of Computer and Information Sciences 5*, 2 (1976), 123-143.