Discount learnability evaluation

Paulo J. Santos
Albert N. Badre

Graphics, Visualizationa & Usability Center
College of Computing
Georgia Institute of Technology
Atlanta GA 30332-0280 USA
Tel: +1 404 8942598
E-mail: {pas, badre}@cc.gatech.edu

ABSTRACT
Learnability evaluation has traditionally required expensive and time-consuming techniques. Practitioners have refrained from performing extended learnability evaluation due to its prohibitive costs. We propose a discount method for evaluation of an interactive system's learnability. Our method is based on automated logging of user actions, detection of user mental chunks, and observation of chunk size as it grows over time with experience. We introduce a model for chunk detection, and present experimental results validating the use of chunk size as an indicator of learnability.

Keywords
Learnability, ease of learning, discount usability, usability evaluation, chunking

INTRODUCTION
One important component of an interactive system's usability is its learnability. Learnability, also known as “ease of learning”, can be defined as a measure of the effort required for a typical user to be able to perform a set of tasks using an interactive system with a predefined level of proficiency. Most authors of usability and interface design publications (including, for example, [12, 16, 18, 22]) recognize the importance of ease of learning in interfaces. Nielsen [12] even claims that “learnability is in some sense the most fundamental usability attribute”.

Despite the consensus that learnability is an important issue in usability, few[1] of those authors discuss at length the issue of learnability evaluation. Some stress the importance of evaluating learnability for the first-time user. Assessment of initial learnability is certainly crucial to the study of a system, but it is by no means a complete study of a system's learnability. It is, at best, a measure of the system's intuitiveness. Measuring intuitiveness, or initial ease of learning, does not provide the analyst with information about the learnability of the system over an extend period of use.

We recognize the difficulties in measuring learnability, and understand that those may be the main reason for the limited discussion of learnability evaluation in most texts. The most significant obstacle to extended learnability studies is probably cost. Observing users over an extended period of time would consume huge resources, particularly observer time. Even if observation could be fully automated (through, for example, the use of event logging), a detailed analysis to study learnability would probably be very costly.

Despite all the difficulties associated with longitudinal studies of learnability, we believe that the evaluation of learnability over an extended period of time is valuable. The initial trend of a learning curve, observed on novice users, is not necessarily an indicator of continued learning. Figure 1 (adapted from [12]) illustrates this concept. A and B represent learning curves for two hypothetical systems. System A favors intuitiveness and fast learning, whereas system B optimizes performance for the expert user. At time t1 in Figure 1, system A would seem to be preferable to system B. But at time t2, system B is clearly the winner. Clearly, in this example, system B is harder to learn at first, but once users overcome the initial obstacles, they learn faster and become more proficient. This example illustrates that an extrapolation of an analysis of short-term performance to infer long-term performance is prone to errors.

Design decisions affecting the learning curve of a system depend on the goals of the system, the target user

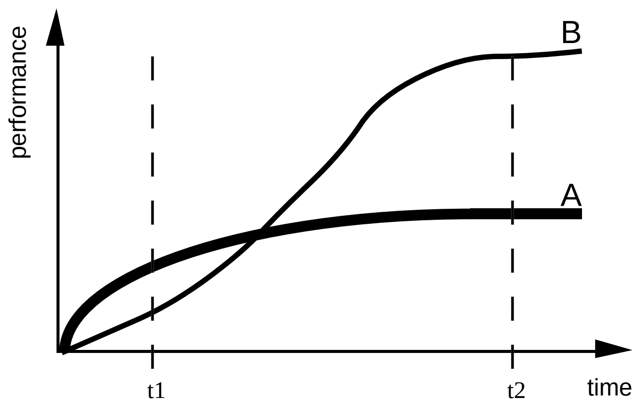

Figure 1 — Two learning curves

1 we are tempted to claim “none” instead of “few”, but we'd be accused of not having reviewed the literature throughly

population, the training costs, the existence of time-critical tasks, and other system design considerations. We are not claiming that either A or B would always be preferable {actually, sometimes it is possible to combine the best of both worlds; configurable and adaptive interfaces are a step in that direction), but are simply stating that one needs to be able to evaluate extended learnability in order to make an informed decision about long term use of the system.

Clearly, there are cases when we need to study extended use of the system, particularly its learnability. Two major techniques have been used to meet this need: cognitive walkthroughs and field observations. Cognitive walkthroughs [10, 14] perform a theoretical evaluation of an interface. Polson and Olson [15] presented some encouraging results of using cognitive walkthroughs to estimate learnability. The disadvantage of the method, however, is that is requires specialists familiar with the cognitive walkthrough methodology to devote a significant amount of time to analyze the interface. Field observations consist of usage data that is collected after the system is deployed and released: user observation, monitoring of customer support calls, and user feedback are all methods that can be used to monitor usage over a long period of time, and estimate learning curves.

All of the existing methods, however, are costly to execute. Cognitive walkthrough requires highly-paid and specialized cognitive scientists to devote time to analyze an interface in detail. And field observations require field trips, encoding of data, and substantial analysis time. Field observations and logging of customer's activities may be further complicated by the need to preserve privacy and confidentiality of the customer's data and use of the system.

Budget overruns and missed deadlines are a serious problem affecting usability evaluation. Often, usability evaluators can not afford to spend the time or money they would like to make a complete usability study. Discount usability techniques [11,13] have become quite popular, because they are more likely to result in a better return on the usability expenditures. The final results of the usability evaluation may not be as good or reliable as compared to those of a more expensive test, but since they are so quick and cheap to execute, any benefits that can be derived from their use come at almost no cost.

We propose a discount method that provides an indication of learnability in a totally automated manner. It is installed to run in the background as the user uses the system, needs little or no configuration to setup, preserves confidentiality of the users data and interaction, and produces numerical results that can be easily compared and plotted. Most importantly, since it can be significantly automated, its cost is minimal.

CHUNKING AND CHUNK DETECTION

Our technique for studying learnability is based on the analysis of user chunks. A chunk is a coherent cognitive unit that the user holds in short term memory while performing a cognitive operation. Chunks and the mechanisms people use for chunking have been studied for the last three decades.

A body of literature, generally referred to as the chunking literature, establishes relationships between cognitive stages and observable behavior. User behaviors have been shown to be intimately related with their cognitive skills. Results such as those found in [1, 2, 5, 8, 21] 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 [8, 17] and tactical decision scenarios [2]. More recently, we have shown that chunk detection is also feasible in human–computer interaction [20], by analyzing the user operations and the pauses between them.

When interacting with a computer, users typically behave according to the acquisition/execution cycle [7]. 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. During this phase, there is typically no physical interaction with the computer. Then they proceed to the execution phase, when a burst of activity occurs as users carry out the plan by interacting with the computer. Once they have reached their goal (or failed to reach it), 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.

Some models have been proposed for the estimation of the execution time of tasks by experts, e.g. [6, 9]. In order to identify user chunks, we use the results of predictive models for execution time. In particular, we start by using a variation of the Keystroke-Level Model [7] as a first approximation. It is simple enough, and fits in nicely with the principle of “discount usability”. Other models may be used as alternatives, and the same principle should still work.

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

- Record all the low level user events (keystrokes, mouse clicks, etc.), using any general purpose logging tool (such as [3, 4]).

- 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.

An illustration of chunking mechanism is provided in Figure 2. The top timeline represents a sequence of user events. By applying the chunking algorithm, we assess chunk boundaries where pauses are too long to be justifiable by a predictive model for execution time, and aggregate the units between boundaries into groups, which we call chunks.

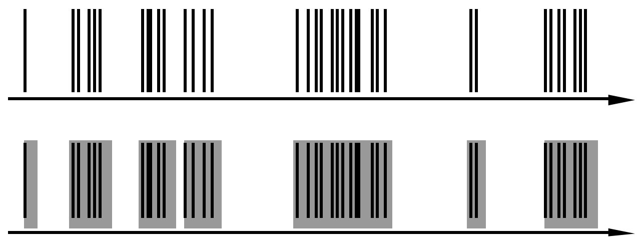
Figure 2 — An example of chunking

In [20] we report the results of an experiment in which we matched chunks detected as described above with user verbal protocols. In that study, we found that the chunks correspond, for the most part, to the execution of user goals.

CHUNKS AS INDICATORS OF LEARNABILITY

One important observed characteristic of chunks is that they grow as users become more experienced. In many other domains, expert subjects have been shown to form larger chunks and more regular chunks than inexperienced users. In the human–computer interaction domain, one would expect this to also be verifiable. More experience users can formulate higher level goals and execute more complex execution plans, which generally require more events (keystrokes or mouse clicks).

We will therefore use chunk size as an indicator of expertise. The variation of chunk size would then be an indicator of learning. We hypothesize that chunk size grows with experience, and that the variation of chunk size is an indicator of learning by the users.

To validate our hypothesis, we conducted an experiment. In the experiment we use an application domain that involves a high degree of strategy and significant mouse-clicking activity: a computer game. The game used will be a version of the game of Minesweeper, which provides for significant learning and use of strategies.

The game of Minesweeper provides many opportunities to learn strategies. After the strategies are learned, the user may apply them. When a strategy involves clicking in multiple cells, the system should be able to detect that a new larger chunk was formed. Users unfamiliar with a particular strategy will reason on cells individually, and probe them one at a time, with significant pauses between them. On the other hand, users that recognize a pattern and have proceduralized a strategy to solve it will probe multiple cells in quick succession. Distinguishing between both types of activity (single or multiple cell probing) is fairly straightforward, and can be done by inter-chunk pauses analysis. We then need to assert that the larger and more complex chunks are observed in the interaction of the more "experienced" user.

In this experiment, we controlled "experience" by influencing learning. Learning was stimulated through tutoring of strategies. Some participants in the experiment were tutored, while others were left to discover the strategies on their own. In pre-experiment pilot studies, we observed that the strategies of the game are not trivial: users required a long time and a large number of trials before they could recognize interesting patterns on the board, and proceduralize complex probing actions. When assisted (tutored), they would immediately recognize a new pattern, understand how it could be derived from smaller patterns that they had already understood, and easily proceduralize the solution of the new pattern. That was a good indication that tutoring might be useful, in this case, to control learning.

Figure 3 provides an illustrative example of this game. The object of the game is to locate all the mines placed in a mine field. The mine field consists of a rectangular board of square cells. In each cell, there could be a mine. The object of the game is to locate all the mines without ever touching one. To probe a cell, the user should click on its square. If there is a mine in that cell, it explodes and the user loses the game. If the cell does not contain a mine, a number will appear, indicating how many mines are in the cells that surround it.

Design

In this experiment, we have two independent variables (tutoring and session number) and one dependent variable (size of chunks.) We control learning of strategies in this experiment through tutoring of patterns and strategies to solve them, and we measure chunk size by the number of actions (mouse clicks) in a chunk. The independent variable "tutoring" has two values: assisted and unassisted learning, with half the participants being assigned randomly to each group. Those in the assisted learning group receive tutoring throughout the experiment, while those in the unassisted learning group receive limited tutoring. The independent variable "session number" has nine values, 1

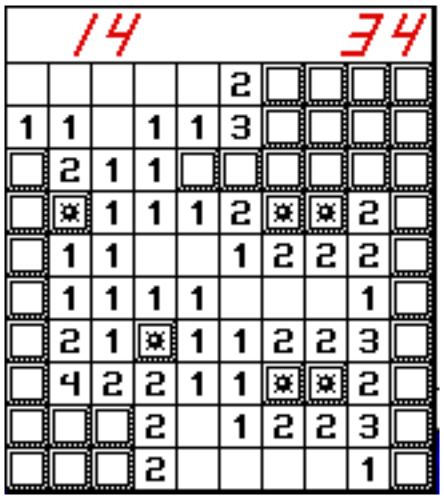

Figure 3— Example of a simple interface to the Minesweeper problem

through 9, corresponding to each of the boards that the participants were asked to solve, in the order thet they were introduced.

To develop the tutoring program, we started by developing and selecting a set of sensible strategies. We decided on a set of nine strategies, and developed a tutoring program designed to teach those strategies in a logical sequence. Lessons in the tutoring program were organized in increasing order of complexity, with each strategy lesson building on the knowledge acquired during the previous lessons. Each lesson consisted of a handout with a diagram illustrating a typical arrangement of known and unknown cells on the board, accompanied by the solution to that particular configuration, with an explanation of the reasoning behind the solution. The participant was then free to discuss the learned strategy with the experimenter, to confirm that each strategy had been understood.

Between lessons, participants used the computer to solve Minesweeper boards. The boards were not random, but had been previously designed to incorporate instances of the strategies that had been learned. Thus, the participants had the opportunity to practice solving boards using the strategies that they had learned. Note that, however, every board was solvable using just the basic rules of the game. Knowledge of the more complex strategies only helped with the performance, but was not required to successfully solve the boards.

Each participant was given a sequence of nine different boards to solve. Every board had the same size (20 by 20 cells), with the number of mines varying between 24 and 37, with an average of 32.7 mines. That corresponds to a mine density range of 6 to 9.25 percent of the board, and an average density of 8.175 percent.

The paragraphs above describe the tutoring and practice scenario used for the participants in the assisted learning group. Participants in this group received two sessions of tutoring at the beginning (before attempting to solve the first board), and then one session of tutoring before each of the next seven boards. No lesson was given between the eighth and ninth boards. Participants in the unassisted learning group received the same two lessons before the first board, but were not given any further assistance. Basically, participants in the unassisted learning group were left to learn strategies and acquire skills by themselves. Presumably, their performance will be no better than the performance of participants in the assisted learning group, and, if the observations in the pilot study are to be considered, their performance should be significantly worse than that of the participants in the assisted learning group.

Participants

Twenty-four participants were used in this experiment. All participants were students at the Georgia Institute of Technology, currently enrolled in undergraduate classes in the School of Psychology. Nine participants were female, and thirteen were male. Ages range from 19 to 34 years, with a median of 21 and an average of 23.4 years. No expertise in computer use was required. However, it was required that participants should not have had exposure to the Minesweeper game (computer or non-computer version) prior to this Experiment. Class credit towards their psychology courses was awarded to every participating participant. Before the first experimental session, each of the 24 participants was randomly assigned to one of the two experimental conditions.

Procedure

Participants were scheduled to participate in 60-minute experimental sessions, during the day time on weekdays. All participants completed the sessions. All but one of the participants finished their participation in the experiment in the allocated time or less; one participant had to voluntarily accept an extension of the session to allow him time to finish all the boards (this participant was in the unassisted learning group.)

Each session started with personal introductions. A good inter-personal relationship between experimenter and participant was important for a successful completion of the tutoring phases of the experiment. The introductions were an important step towards establishing a good relationship, as they fostered confidence and put the participant at ease. Participants were then asked to read and sign an informed consent form, authorizing their participation in the experiment. The formal part of the experiment session began with a reading of the instructions, including and outline of the experimental procedure and complete game rules. Participants were told that their goal was to complete as many boards as possible, in the shortest amount of time. The speed factor was introduced to encourage participants to work fast, leading them to proceduralize their chunks.

The board using during the experiment had the visual presentation similar to that presented in Figure 3. At the top of the screen is a box displaying a number that indicates how many mines are left to be uncovered. To the user, that may be seen as the current game “score”. The object is to take the score down to zero by locating all the mines. In addition, the system displays help messages such as “you may begin” or “use left button to probe or right to mark” at the top, to the right of the “score” indicator. The remaining useful screen space is occupied by the game board, with 20 by 20 cells. Every cell was initially white, indicating that no information about the cell was known. As each cell was probed (by clicking on it using the left mouse button), its background would become black. If the probed cell contained a mine, the mine would “explode”, revealing the location of every mine on the board and terminating the game with a failure. If, on the other hand, a probed cell did not contain a mine, the number of mines in its adjacent cells would be revealed. If, however, the there was no mine to be found in any of the adjacent cells, the cells probed would simply turn black without the indication of any number (implicitly zero), and the system would automatically probe all the neighboring cells. It would then continue to recursively probe neighboring cells of additional cells that had no mines in their neighbors. This automatic probing of neighboring cells was done only for this trivial case in order to save time during the experimental session. Grid lines on the board were drawn in orange, to clearly distinguish them from all the black and white features on the board.

To perform the two operations on each cell (probe or mark), the left and right mouse buttons were used. Since both actions (probe and mark) use equivalent interaction techniques, there should be no effect of selection technique on performance. The use of these equivalent interaction techniques should eliminate a possible effect on performance of executing the operations.

A Minesweeper program was developed specifically for this experiment. It was developed using the C language on a UNIX system, with the interface being done by use of the X Windows System. The program read board configuration data from a layout file, and internally recorded every event as the user clicked to probe and mark cells. The user events and internal system state were recorded to a file during the interaction. A timestamp was associated with every user event. Timestamps were recorded in milliseconds relative to the beginning of the session (time 0) and are believed to be accurate to 1/60th of a second. The experiment was executed on a SparcStation II running SunOS operating system (a flavor of UNIX) and X Windows. No knowledge of UNIX or X Windows was required of the user. A pre-arranged script for each session automated the session, and the user only had to solve the sequence of boards by using the mouse to probe and mark cells.

For the tutoring session, participant and experimenter sat at a desk to the side of the computer desk, and together went over the strategy being introduced in the lesson. The participant was first given the opportunity to read and analyze the page containing the lesson material, and then participant and experimenter discussed the strategy, and how one would reason to get to that strategy.

The interaction with the computer occurred with the participant sitting in front of the computer, and the experimenter apparently retreated from the scene. The experimenter stayed in the room during the interaction portions of the sessions, but never approached the participant or the computer to intervene or to closely observe the interaction.

At the end of each session, users were asked to fill out a demographic questionnaire, and were given class credit for their participation in the Experiment.

Analysis

Comparing inexperienced and experienced users (in our case, unassisted and assisted participants, respectively), we should observe that experienced participants form larger chunks (more events peer chunk) than inexperienced participants.

There will certainly be other differences (for example, success rate, total time for completion of each problem), but analyzing these differences would not necessarily reveal information directly related to chunks.

To analyze the results of this experiment, we compare the chunk size of the chunks detected for the two experimental groups. If our hypothesis that our chunk detection algorithm really detects chunks is true, then we should observe characteristics of chunks in the units detected by our algorithm. In this experiment, we would expect the following results:

- <u>Chunks should grow as users become more experienced</u>. In this experiment, we will verify that there is a positive and significant positive effect of "session number" on chunk size.

- <u>More experienced users will form larger chunks than less experienced users</u>. In this experiment, we will verify that there is a significant positive effect of the "tutoring program" (which directly relates to learning speed) on chunk size. Such an effect should occur only after some sessions of tutoring. In particular, there should be no difference between groups on their performance in the first session (since both groups had received the same tutoring up to that point), but the difference should manifest itself in later sessions.

To perform the analysis that would lead to answers to the above questions, we analyzed the data as a three-factor two-stage nested design. Three factors were considered. One independent factor was "session number", with nine levels (1 through 9.) Another independent factor was "tutoring program" with two levels (assisted or unassisted learning). Finally, a third factor, nested within "tutoring program", was "participant", with twelve levels for each level of the "tutoring program" factor. We performed an analysis of variance for the above design, and identified the sources of variance in the experiment.

Results

We executed 24 sessions (12 in each group). In each session, participants solved or attempted to solve nine Minesweeper boards, for a total of 216 boards across all sessions. Across all those sessions, a total of 7,485 user events were generated. Each event corresponds to one user action on the board ("mark" or "probe" a cell.)

After collecting the data from all the sessions, we executed our chunk detection algorithm on all the data sets. Running the data through the chunk detection algorithm resulted in a total of 4,184 chunks. Just over half of all the chunks (2,155, or 51.5% of the total) consisted of single events. The largest chunk recorded contained seven events. Figure 4 shows a distribution of chunk size, as a percentage of total number of chunks.

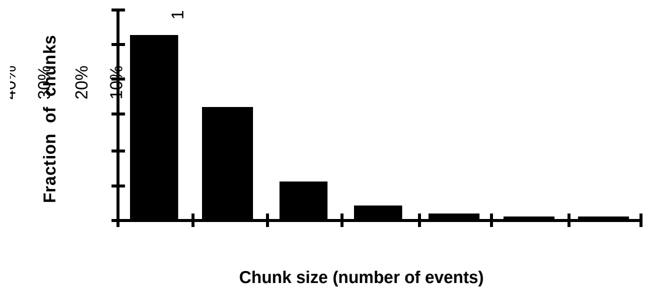

Figure 4 — Distribution of chunk size (number of events per chunk) across all sessions

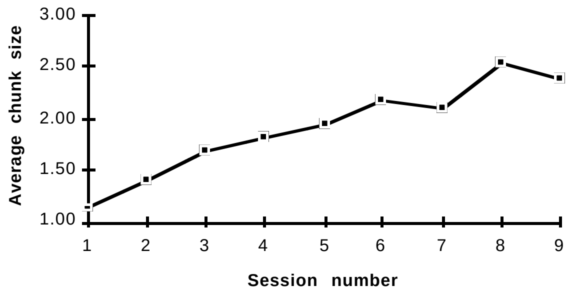

Figure 5 — Chunk size versus session number

We then performed an analysis of variance on the data. From this analysis we conclude that there is a strong effect of tutoring on chunk size. Although there is no strong effect of session alone, there is a significant effect of the interaction of session with tutoring. Finally, there is also a significant effect of participant (within tutoring). All of these are significant at the one percent level. In addition, the interaction of session and participant is significant at the five percent level. (Detailed analysis results are available in [19].)

The purpose of this experiment was to test that chunk size grows with experience. Experience should be influenced by two factors in this experiment: the number of boards previously solved, and the tutoring program. Table 1 presents average chunk size, across all 24 participants, for each session. The same data is illustrated graphically in Figure 5.

Session number	Average chunk size
1	1.13
2	1.39
3	1.66
4	1.80
5	1.94
6	2.18
7	2.10
8	2.52
9	2.38

Table 1 — Chunk size versus session number

There is a clear indication that chunk size increases with session number. From the analysis of variance we see that chunk size is affected by several factors that contribute to experience and skill: tutoring and the interaction of tutoring and session.

The analysis of variance also shows that tutoring strongly affects the chunk size. Table 2 presents the data collected to verify that there is a significant positive effect of tutoring on chunk size. Participants in the assisted learning group constructed larger chunks faster than participants in the unassisted learning group. Participants in the assisted learning group were forming chunks averaging two events by session 3, while participants in the unassisted learning group took until session 8 to reach that level.

Session	Assisted	Unassisted
1	1.13	1.14
2	1.49	1.30
3	1.91	1.43
4	2.30	1.46
5	2.60	1.57
6	2.59	1.64
7	2.39	1.81
8	3.38	1.99
9	3.31	1.96

Table 2 — Chunk size versus session number versus tutoring

The analysis of variance also shows that tutoring strongly affects the chunk size. Table 3 clearly stresses that the effect is a strong positive one, by showing average chunk size for each participant during the first and last sessions. No data is shown for the last session of participants 2 and 24 because both of these participants hit a mine on their first probe of the board.

The analysis of variance had not shown any significant effect of session on chunk size. There is, however, a strong effect of the interaction of session and tutoring. This suggests that the learning, if any, of non-assisted participants is confounded with experimental error and individual differences. Clear examples of that lack or learning were exhibited, for example, by participants 6, 15 and 18 (all in the unassisted learning group).

Participant	Session 1	Session 9
1	1.11	3.00
2	1.12	
3	1.11	2.24
4	1.14	1.73
5	1.11	5.00
6	1.05	1.25
7	1.10	3.59
8	1.14	2.74
9	1.08	1.76
10	1.10	4.06
11	1.17	3.40
12	1.19	3.25
13	1.19	2.20
14	1.13	1.80
15	1.14	1.00
16	1.21	3.50
17	1.00	3.00
18	1.18	1.25
19	1.12	2.12
20	1.15	2.09
21	1.07	1.50
22	1.11	2.00
23	1.30	2.05
24	1.15	

Table 3 — Chunk size for each participant, first and last session

Discussion

The results of this experiment strongly support the claim that the groups of events identified by our chunk detection algorithm share characteristics with chunks. Specifically, the size of events recorded grow with experience and skill. This is the same characteristics that has been shown of chunks.

Individual differences were strongly noticed in this experiment. We had observed anecdotal evidence of significant difference between individuals during the experimental sessions. Some participants were good logical reasoners, some were good at recognizing patterns, some were fast learners, … Others, not quite so. The analysis of the data confirmed that individual differences significantly affect the chunk sizes. This is a very expected result, as people have been shown to differ greatly in tasks that involve cognitive skills, learning, and complex reasoning. However, the object of this experiment was not to discern how individual differences affect performance at Minesweeper. Therefore, we simply confirm that individual differences are significant, but we do not delve into an analysis of the causes or consequences of those differences. It is important, however, to state that this experiment shows that our chunk detection algorithm appears to provides reliable results across a wide variety of individuals, and that chunk size seems to be a consistently good indicator of expertise.

CONCLUSIONS AND FUTURE WORK

An analysis of chunk size over time provides an indication of user expertise. Since expertise is a direct result of learning, the variation of chunk size over time is an indicator of learnability. Figure 5, plotting the experimental results of chunk size over time, may be viewed as an approximate illustration of a learning curve.

Soon, in cooperation with a major software company, we plan to instrument beta versions of software to record keystroke information from the usage of beta testers. We will then analyze chunk sizes from the logs, and have some indicators of learnability available for the company before the final version of the product is released.

REFERENCES

1. 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.

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

3. Badre, A.N., Guzdial, M., Hudson, S.E., and Santos, P.J., *A user interface evaluation environment using synchronized video, visualizations and event trace data*, Journal of Software Quality, **4**, pp. 101-113, 1995.

4. Badre, A.N. and Santos, P.J., *CHIME: a knowledge-based computer–human interaction monitoring engine*, Technical report GIT/GVU-91-06, Graphics, Visualization and Usability Center, Georgia Institute of Technology, Atlanta, Georgia, 1991.

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

6. 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), pp. 32-74.

7. 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), pp. 396-410.

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

9. 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, pp. 365-394.

10. Lewis, C., Polson, P., Wharton, C., and Rieman, J., *Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces*, in *Proceedings of ACM CHI'90 Conference on Human Factors in Computing Systems* (Seattle, WA, April 1–5, 1990), ACM Press, pp. 235–242.

11. Nielsen, J., *Usability engineering at a discount*, in Salvendy, G. and Smith, M.J. (eds.), *Designing and Using Human-Computer Interfaces and Knowledge Based Systems*, Elsevier Science Publishers, Amsterdam, 1989, pp. 394-401.

12. Nielsen, J., *Usability engineering*, Academic Press, Boston, 1993.

13. Nielsen, J., Applying discount usability engineering, IEEE Software 12(1), Jan. 1995, pp. 98–100.

14. Polson, P., Lewis, C., Rieman, J., and Whaton, C., Cognitive walkthroughs: a method for theory-based evaluation of user interfaces, International Journal of Man–Machine Studies, 36(5), May 1992, pp. 741-773

15. Polson, P.G. and Olson, J.S., *An approximate method for estimating total learning time: an example of a usability inspection method derived from cost/benefit considerations*. Paper presented at Human–Computer Interaction Consortium Annual Winter Workshop (Pittsburgh, PA, Jan. 25–28, 1992).

16. Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., Carey, T., *Human–Computer Interaction*, Addison-Wesley, Wokingham, England, 1994.

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

18. Rubin, J., *Handbook of usability testing: Hot to plan, design, and conduct effective tests*, John Wiley & Sons, New York, 1994.

19. Santos, P.J., *Automatic detection of user transitionality by analysis of interaction*, doctoral dissertation, Georgia Institute of Technology, 1995.

20. Santos, P.J. and Badre, A.N., *Automatic chunk detection in human–computer interaction*, in *Proceedings of the Workshop on Advanced Visual Interfaces AVI'94* (Bari, Italy, June 1–4, 1994), ACM Press, New York, pp. 69–77.

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

22. Shneiderman, B., *Designing the user interface: strategies for effective human–computer interaction*, 2nd edition, Addison-Wesley, Reading, Massachusetts, 1992

23. Wharton, C., Rieman, J., Lewis, C., Polson, P., *The cognitive walkthrough: a practitioner's guide*, in Nielsen, J. and Mack, R.L. (eds.), *Usability inspection methods*, Wiley, New York, 1994, pp. 105–140.