Explorations in the Parameter Space of a Model Fit to Individual Subjects' Strategies while Learning from Instructions

Margaret M. Recker*

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332
(404) 894-9218
email: mimi@cc.gatech.edu

Abstract

In earlier work, we presented results from an empirical study that examined subjects' learning and browsing strategies as they explained instructional materials to themselves that were contained in a hypertext-based instructional environment. We developed a Soar model that, through parameter manipulation, simulated the strategies of each individual subject in the study. In this paper, we explore the parameters of these simulations and contribute several new results. First, we show that a relatively small proportion of strategies captured a large percentage of subjects' interaction behaviors, suggesting that subjects' approach to the learning task shared some underlying strategic commonalities. Second, we show that lower performing subjects employed a high proportion of working memory intensive strategies, which may have partially accounted for their inferior performance. Third, clusters of subjects identified through parameters analyses continued to exhibit similar behaviors during subsequent problem solving, suggesting that the clusters corresponded to genuine strategy classes. Furthermore, these clusters appeared to represent general learning and browsing strategies that were, in some sense, adaptive to the task.

Introduction

Possessing a diverse, flexible set of problem solving strategies is a well-recognized hallmark of an effective problem solver. Clearly, a person who can avail him/herself to more than one problem solving strategy has more potential power than someone possessing only one method. The same kind of argument applies to learning: A learner with a repertoire of learning strategies – and who can effectively select and apply them – should better profit during learning situations.

Identifying and understanding the various strategies that are drawn upon during learning tasks is receiving increased attention in both artificial intelligence and cognitive science. For example, in AI, recent research has focussed on building multi-strategy learning systems (e.g. Ram & Cox, 1992). Examples in cognitive science include studies that have focussed on understanding the strategies that learners bring to bear while explaining instructional materials to themselves, in a learning strategy called *self-explanation*. Prior research has found considerable individual differences in learners' self-explanation strategies, and these appear to significantly affect their initial understanding and their subsequent problem solving performance (Chi, Bassok, Lewis, Reimann, & Glaser, 1989; Pirolli & Recker, in press).

In earlier work, Recker & Pirolli (in press) presented a computational model that captured individual differences in learners' self-explanation strategies. The model, called SURF, was based on results from a study in which computer-based instructional situations were manipulated in order to measure their effects on learners' self-explanation strategies. The modelling approach involved coupling simulations of subjects' interface interactions strategies with opportunities for action supported by the interfaces of the instructional environments. By setting parameters in the model, it was fit to the learning behaviors of individual subjects. Specifically, for each subject, the corresponding simulation was required to perform, in exactly the same temporal order, the subject's mouse actions and self-explanations.

In this paper, we present analyses of the parameters used by the model to fit individual subject data. As we will discuss, these parameters can be seen as representing the subjects' learning strategies as they interacted with the computer-based instructional systems. In the next section of the paper, we describe the SURF model and review the empirical results that provided its motivation. We then present analyses of the model parameters that were used to fit subject data.

The SURF Model

Empirical Motivation

We recently conducted a study in which the structure of instructional materials was manipulated in order to examine effects on learners' self-explanations strategies (Recker & Pirolli, in press). In this study, 16 subjects went through five lessons on programming in Lisp. Each lesson had two parts: studying instructional material

*This work was conducted while the author was at the University of California, Berkeley, and was supported by NSF under contract IRI-9001233 to Peter Pirolli and by a University of California Regents' Dissertation Fellowship.

(learning), followed by programming (problem solving), using an intelligent tutoring system for Lisp, the Lisp Tutor (Anderson, Boyle, Corbett, & Lewis, 1990). For the target lesson, the lesson on recursion, two sets of computer-based instructions were developed. Subjects were randomly assigned to one of the two environments to learn about the concepts of recursion prior to programming recursion with the Lisp Tutor. The first environment was embedded in a hypertext system, which allowed subjects to browse the material in a non-linear way. In addition, the instructions were annotated with explanatory elaborations that subjects could choose to view if they were unable to generate their own self-explanations. Subjects who were able to generate their own self-explanations could choose to ignore these additional textual elaborations. A second, control instructional environment was also implemented, which mirrored more standard, linearly structured instruction. Subjects navigated through both environments by clicking on buttons. These mouse actions provided additional data on subjects' strategies for learning from instruction.

When we contrasted subjects' performance while programming with the Tutor, we did not find any significant differences in outcome between subjects using the hypertext-based instructional system and those in the control. However, we did find a significant aptitude-treatment interaction. Post-hoc analyses showed that the higher-ability subjects (those that performed well in the pre-intervention lessons) in the hypertext condition made significantly less errors while programming than the low-ability subjects. Subjects in the control condition did not show significant ability-based differences.

Model Overview

Data from the empirical study formed the motivation for a computational model, called SURF (Strategies for Understanding Recursive Functions). SURF was implemented within the Soar architecture (Laird, Rosenbloom, & Newell, 1987). Soar is a production system architecture in which problem solving is carried out by search through problem space in order to achieve particular goals. Soar also includes an experience-based learning mechanism, called *chunking*, which summarizes problem solving experiences into a more efficient form.

In the interest of space, we can only present an overview of the SURF model. In addition, since we will only discuss the simulations of the eight subjects in the hypertext condition, the model of the control condition will not be presented. More details can be found elsewhere (Recker & Pirolli, in press).

The primary goal of the SURF model was to model the learning behavior of individual subjects in terms of two criteria. The first criterion required that *every* mouse clicking action by all subjects be simulated in the exact order that it occurred. This formed the *fine* modelling criterion. Subjects' self-explanations formed the secondary, *coarse* modelling criterion. This meant that subjects' self-explanations were modelled at a rough level, in the sense that their exact natural language statements were not simulated. More specifically, at each screen in the instructional environments, a subject could attempt to self-explain the instruction. The subjects' verbal protocols were consulted to determine when domain-related self-explanations were exhibited. At each of these instances, the corresponding simulation would apply what was called the *comprehension* operator. The application of the comprehension operator resulted in the creation of chunks, representing newly acquired knowledge. Thus, the SURF model, though not yet a complete model, focussed on exactly capturing the temporal sequence of subjects' interface interactions and domain-related self-explanations.

In the SURF model, subject-environment interactions are modelled as three components: (1) a simulation of the instructional interface, (2) a model of the space of possible interactions, and (3) simulations of individual subjects' interaction strategies.

The interface of the instructional environment was a Lisp simulation of the buttons and instructions that were displayed in each screen context. These buttons and instruction snippets represented opportunities for actions. The last two components were Soar simulations and are described in the following sections.

Modelling Interactions

We have defined an interaction as occurring between environmental features (the interface) and subjects' learning strategies and prior knowledge (jointly called capabilities). Taking environment and learner factors together defined a space of interaction possibilities. In order to build a model of this space separately from the influence of individual learners' capabilities, it was implemented such that it included the entire space of possibilities. In practice, when the model of interactions was loaded with a particular learner profile, a subset of this space was explored.

Learners' Capabilities

The last component of the model involved simulating individual learners' capabilities. A set of production rules was created for each subject, called the *profile*, which represented each subject's learning strategies and prior knowledge. Each subject's profile was implemented such that when it was loaded in with the interface and interaction models, the resulting Soar run would fit that subject's behavior. Recall that the fit had to meet two criteria: the *fine* criterion required that every mouse clicking action had to be captured in the order that it occurred and the *coarse* criterion modelled subjects' self-explanations at a rough grain-size.

Two kinds of methods existed for modelling learners' capabilities. First, a set of production rules represented the learner's prior knowledge that was used while generating a self-explanation. A second set of production rules represented how the learner selected among possible available actions (or operators). In Soar, the de-

sirability or acceptability of possible alternatives is described in terms of a fixed language of *preferences* (e.g., best, better, reject). Preference knowledge is stored in production rules, and is added during the problem solving decision cycle. In SURF, preference productions are used to express the value of available operators (e.g., selecting a particular button is desirable) in order to simulate a subject's action in the exact order that these actions occurred. Since preference productions deliberately choose among available operators (and thus are knowledge about knowledge), they can be seen as representing strategic or metacognitive knowledge.

For example, one subject always exhibited a preference for viewing examples when they were available. That is, he showed a consistent preference for selecting the "See Example" button over all other buttons. This kind of strategy was implemented by the following preference production rule (shown in pseudo-Soar notation):

```
If the context involves reading instruction
 And the instruction is contained on a screen
 And the screen contains buttons
 And a button is labelled ''See Example''
 And the button can be selected
Then
 This button selection is the best
```

Another kind of preference might express the high desirability of self-explaining instructional examples. This would translate into a preference for selecting the comprehension operator when an example is displayed on the screen.

Comparing subjects and simulations

A profile for each subject was implemented according to the modelling criteria and then run. When run, each simulation would learn different amounts and kinds of chunks, representing new knowledge gained from the instruction. Presumably, a simulation that acquired many chunks represented good learning from instruction. We would then expect that the corresponding subject would exhibit successful problem solving performance and thus record a low number of programming errors. In fact, a Spearman rank order correlation between the number of chunks created and the mean number of errors on the recursion lesson showed a significant negative relationship between number of chunks and error rates (rho = -.76; $p < .05$).

An Ideal Strategy

This modelling framework allowed us to run experimental simulations. A profile was implemented that performed an "ideal" walk through the instruction. This simulation viewed every screen of instruction and applied the *comprehension* operator in every possible context. Its profile contained nine preference production rules and did not represent an actual subject.

Parameter Analyses

Essentially, SURF is a model of *when* to self-explain, and less a model of *how* to self-explain. The preference production rules contained in the subjects' profiles provided search control knowledge that modelled its corresponding subject's mouse clicking actions and (roughly) self-explanations, in the exact temporal order that they occurred. As such, the preference productions represented subjects' strategic knowledge for explicitly choosing among available actions (i.e., their learning strategies). They can also be seen as parameters used to fit a model of individual differences. They thus provide a tangible means of measuring the space of individual differences in strategy use. In this section, we present several analyses based on the set of preference productions contained in each subject's profile.

Preference Productions

Table 1 shows the number of preference productions contained in subjects' profile. The table is sorted in ascending order by subjects' mean number of errors while programming with the Lisp Tutor.

Overall, 149 preference productions were required to simulate the behaviors of the eight subjects in the hypertext learning condition, according to the modelling criteria. At first glance, this might seem like a large number, reflecting a random or highly variable behavior on the part of the subjects. However, if we exclude preference productions that only occur once, we are left with a total of 26 preference productions. Furthermore, this core set accounts for a large proportion of the preference productions contained in the profiles. As can be seen in Table 1, the percent of coverage provided by the core set of productions for each of the profiles is above 65% for all but two subjects.

The fact that the learning strategies of subjects could be accounted for fairly well with a small number of preference productions suggests that, overall, subjects were approaching the task with a non-random and common underlying set of strategies. This result is not surprising if we consider the fact that learning from instruction is a well-practiced activity for students. The two subjects with a low degree of coverage will be returned to later.

Working Memory Load

We conjectured that the low-performing subjects may have been overwhelmed by the complexity of the hypertext-based environment. In order to examine this hypothesis, we identified the preference productions in the core set that made high working-memory demands. Operationally, these were preference productions that contained at least on condition element that consulted the state context. We assumed that high working memory preference rules imposed a high cognitive load, which may have interfered with learning effectiveness (Sweller, 1988) and hence reflect a subject with poor programming performance.

Subject	Prods	%Shared	%High W.Mem.
Ideal	9	77.77	14.28
MH74	7	71.42	0
HL63	16	68.75	18.16
CC69	31	25.80	12.50
LP68	23	52.17	33.33
KB70	13	84.61	40.00
MW55	19	73.68	57.14
WP58	19	84.21	50.00
SP73	12	83.33	20.00
Mean	15.55	69.08	27.26

Table 1: The number of preference productions per profile, the percent coverage provided by productions in the core set, and the percentage of working memory intensive productions in a profile. The table is sorted in increasing order of subjects' mean number of errors while programming with the Lisp Tutor.

The preference production presented in the previous section is an example of a production that was not considered working memory intensive in that it did not refer to the state context. However, the following example production was considered working memory intensive. The production required that the state context keep track of the instructional screens that have been viewed.

```
If the context involves reading instruction
 And the instruction is contained on a screen
 And the screen has previously been viewed
 And the screen contains a button
 And the button can be selected
 And the button is labelled ''See Example''
Then
 This button selection is the best
```

The following is another example of a working memory intensive preference production. This production rule required that the state context keep track of all operators that have been applied.

```
If the context involves reading instruction
 And an operator is available
 And the operator has previously been applied
Then
 Reject the operator
```

The last column in Table 1 shows the percentage of preference productions in subjects' profile that were considered working memory intensive. As can be seen, except for the lowest performing subject, the percentage generally increases as the performance of subjects decreases, suggesting that high working memory strategies interfered with learning effectiveness.

Adaptive Strategies

Maximal generation of self-explanations during the learning phase is not necessarily the most effective or efficient means for achieving understanding. Subtle tradeoffs exist the costs of elaborating the instruction and the gains resulting from knowledge acquisition. These trade-offs also probably interact with learners' prior knowledge. Moreover, we can assume that different instructional resources have different gain functions associated with them (Pirolli, 1993). For instance, examples are often more instructive than plain text, that is, they have higher information gain functions. This is a fact that many learners seem to have caught onto: novices' preferences for examples in the early phases of learning a new domain is a robust finding in the literature (LeFevre & Dixon, 1986; Pirolli & Anderson, 1985). Furthermore, these gain functions are negatively accelerating and thus show diminishing returns with continued elaborations (Pirolli & Recker, in press).

Therefore, the learner must decide how to allocate time and self-explanation effort on different kinds of instructional resources with the goal of maximizing (within computational constraints) understanding. If we view the learner as adapted to the task, these kinds of decisions are cached out in learners' overall strategic approach to the task[1].

With this simple model in mind, we can identify general strategy classes that are attempting to adapt to the demands of this learning task. In the first strategy class, gains are maximized during understanding by expending time and effort to extract information from the instruction in order to be prepared for problem solving. In the second strategy class, costs are minimized by minimizing cognitive effort. Here, the structure of the environment is used to suggest actions, in a strategy we call *interface-driven*. In the third strategy class, costs and gains are balanced. The instruction is first skimmed; then during problem solving, after task goals have become more concrete, the instruction can be restudied with these as constraints. Evidence for the existence of such strategy classes is presented in the next section.

Strategy Clusters

The subjects' profiles can be used to identify strategy clusters among different subjects. In order to accomplish this, a matrix was constructed where the rows of the matrix represented subjects. The columns represented each of the preference productions contained in the profiles. The profile for the "ideal" simulation was also included in this matrix. For each preference production, a "1" was entered in subjects' row if their profile contained that production. If the production was not in a subject's profile, the corresponding matrix entry was "0." The resulting matrix was 9 (8 subjects plus the "ideal" profile) by 26 (core preference productions).

Based on this matrix, a hierarchical cluster analysis was performed, using the normalized percent disagreement as the distance metric. Figure 1 shows similar profiles as nodes grouped on branches that begin toward the left side of the figure. As the branches extend

[1]See (Pirolli, 1993) for a discussion of rational analysis (Anderson, 1991) applied to sense-making tasks.

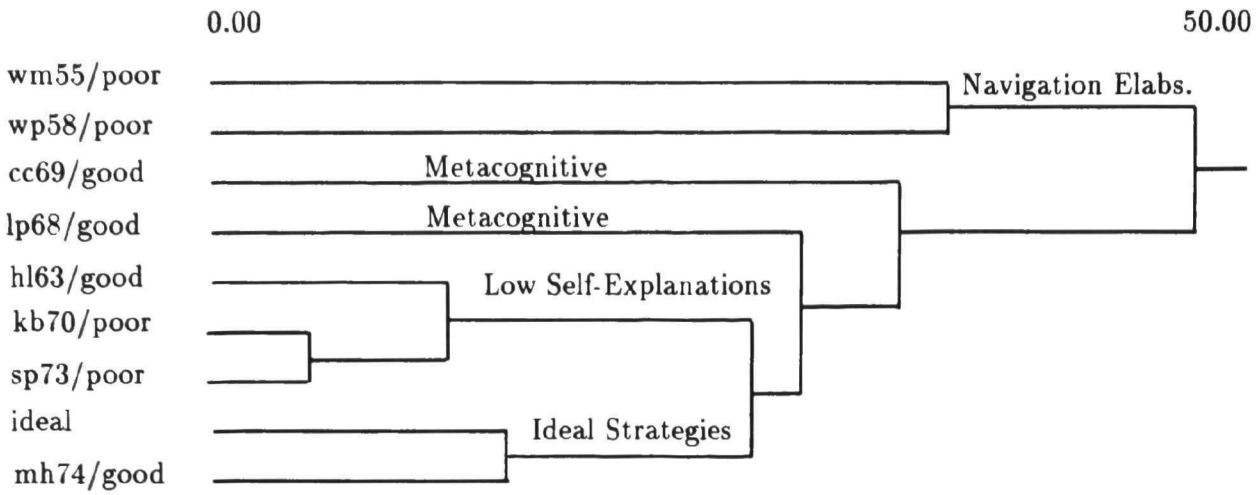

Figure 1: Hierarchical cluster analysis of subjects' profiles.

to the right, toward the origin of the tree, nodes on these branches are increasingly dissimilar.

Do these clusters represent subjects who share an underlying common strategy, and do they map onto the previously identified strategy classes? To examine these questions, we considered subjects whose profiles were grouped within clusters in order to determine if they exhibited similar behaviors during the programming (problem solving) phase.

As can be seen in Figure 1, the cluster analysis grouped together the highest performing subject (MH74) and the "ideal" profile. Because the "ideal" profile was intended to represent maximal gains during the learning phase, this cluster was assumed to fit within the first strategy class. Accordingly, we would expect minimal reliance on the instruction during problem solving. In fact, this subject did not rely at all on the instruction during problem solving.

In the second strategy class, cognitive costs are minimized. We defined this as learners whose actions were decided on the basis of the features present on the interface, which we dubbed the *interface-driven* strategy. In order to determine subjects who may have fit into this category, we examined the verbal protocols of subjects during the instruction-studying phase to identify those subjects who made a large number of elaborative statements about navigation. These kinds of statements suggest that a subject was driven more by features of the interface. That is, the self-explanation process was decided on the basis of buttons available on the screen.

The verbal protocol analysis revealed two subjects, MW55 and WP58, who made a large number of navigation-related elaborations. The profiles of these subjects were also grouped together by the cluster analysis. In order to determine if these subjects genuinely reflected the *interface-driven* strategy class, we examined their behavior during programming to record the extent that they relied upon the Tutor's interface. Specifically, we counted the number of times that subjects used a special Lisp Tutor menu feature that allowed them to ask the Tutor to perform the next problem solving step. In a striking difference, the two subjects defined as *interface-driven* used this feature a total of 15 times, while the remaining six subjects only used it a total of 3 times. Thus, it appears that the interface-driven subjects continued to employ this strategy during subsequent problem solving, suggesting that it corresponded to a genuine strategy class.

Furthermore, we note that the profile of these two subjects recorded the highest proportion of working memory intensive preference productions. At first blush, this may seem contradictory since it could be argued that in the interface-driven strategy, the interface itself acts as an external memory. In short, buttons serve as external cues for actions, and hence should minimize memory demands. However, for this kind of strategy to be effective, the external environment needs to display regularities. As we argued elsewhere, the interface of the hypertext-based instructional environment was somewhat idiosyncratic and probably imposed a significant cognitive load on its users (Recker & Pirolli, in press). Thus, in the case of this hypertext interface, offloading computations onto the environment had unanticipated negative consequences.

The third general strategy involved skimming the instruction first, then reviewing the instruction during problem solving when task goals are concrete. The cluster analysis suggested two subjects that fit within this strategy class, KB70 and SP73. These subjects generated the lowest number of self-explanations during the learning phase. As expected, these subjects relied heavily on the instruction while solving the first programming problem. Furthermore, their search of the instruction was not very directed, since the content of the instruction was less familiar to them. This suggests that subjects did not gain much knowledge during the learn-

ing phase and the difficulties they encountered during the first problem caused them to re-study the instruction.

Finally, two subjects, LP68 and CC69, did not fit within any well-defined strategy class. Note also that the profiles of these two subjects showed the lowest coverage by the core set of preference productions. Their profiles also contained the highest number of preference productions. Inspection of these subjects' protocols revealed that they made a high number of monitoring and strategy-related elaborations and thus seemed very metacognitively-driven in their self-explanation strategies. As reflected in the high number of preference productions and low coverage by the core set (see Table 1), such a strategy was not adequately accounted for in the present modelling framework. While these subjects may have been employing adaptive strategies, the SURF model and parameter analyses failed to captures these.

Discussion

Much modelling work in cognitive science has focussed on constructing one normative model of a behavior, typically representing a single or a composite subject. However, in this paper, we deliberately concentrated on strategy differences between subjects. This focus allowed an analysis of the parameters involved in fitting a model to individual differences in strategy use, and to make several new contributions toward understanding subjects' strategies for learning in a computer-based instructional context.

The analyses presented suggest that, overall, subjects' approach to the learning task shared some strategic commonalities. This is not surprising when we consider that, although the technology is novel, learning from instruction is a familiar activity for most students.

We showed that lower performing subjects employed a high proportion of working memory intensive strategies. The use of working memory intensive strategies perhaps interfered with learning effectiveness, and therefore partially accounted for the aptitude-treatment interaction reported in the empirical results.

Finally, clusters of subjects identified by parameters analysis appeared to exhibit similar behavior types during subsequent problem solving, suggesting that the clusters correspond to valid strategy classes. We also argued that subjects within clusters may be attempting to use strategies that are adaptive to the overall task demands.

However, we note that while the strategies may be adaptive, they are not necessarily *optimal*. Instead, they may represent local maxima within the space of strategies. Any claims about the optimality of a particular approach rest upon an analysis of the gain functions associated with different instructional resources, the possible interaction of these functions with learners' prior knowledge, and their elaboration costs.

This approach differs from another model of self-explanation, the Cascade model (VanLehn & Jones, in press). Cascade modelled the self-explanation strategies and subsequent problem solving of individual subjects' in the physics study of Chi et al. (1989). In the Cascade model, the primary focus was on modelling the content of subjects' self-explanations. While clearly SURF needs to better incorporate *how* to self-explain, this paper took some steps to answering *when* to self-explain and *why*.

Reference

Anderson, J. R. (1991). The place of cognitive architectures in a rational analysis. In VanLehn, K. (Ed.), *Architectures for Intelligence*, pp. 1–24. Erlbaum, Hillsdale, NJ.

Anderson, J., Boyle, C., Corbett, A., & Lewis, M. (1990). Cognitive modeling and intelligent tutoring. *Artificial Intelligence*, *42*(1), 7–49.

Chi, M., Bassok, M., Lewis, M., Reimann, P., & Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, *13*, 145–182.

Laird, J., Rosenbloom, P., & Newell, A. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, *33*, 1–64.

LeFevre, J., & Dixon, P. (1986). Do written instructions need examples? *Cognition and Instruction*, *3*, 1–30.

Pirolli, P. (1993). Information foraging: A new view on problems in human-computer interaction. Paper Presented at Human Computer Interaction Consortium Winter Workshop, Georgia Tech, Atlanta, GA.

Pirolli, P., & Anderson, J. (1985). The role of learning from examples in the acquisition of recursive programming skills. *Canadian Journal of Psychology*, *39*(2), 240–272.

Pirolli, P., & Recker, M. (in press). Learning strategies and transfer in the domain of programming. *Cognition and Instruction.*

Ram, A., & Cox, M. (1992). Introspective reasoning using meta-explanation for multistrategy learning. In Michalski, R., & Tecuci, G. (Eds.), *Machine Learning: A Multi-Strategy Approach*, Vol. IV. Morgan Kaufman Publishers, Los Altos, CA.

Recker, M., & Pirolli, P (in press). Modelling individual differences in students' learning strategies. *Journal of the Learning Sciences.*

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, *12*, 257–285.

VanLehn, K., & Jones, R. (in press). Learning by explaining examples to oneself: A computational model. In Meyrowitz, A., & Chipman, S. (Eds.), *Cognitive models of complex learning.* Kluwer Academic Press, Norwell, MA.