A MODEL OF EXPERT DESIGN

Beth Adelson, David Littman, and Elliot Soloway

Cognition and Programming Project
Department of Computer Science
Yale University
New Haven, Connecticut 06520

1. Motivation and Goals[1]

In this paper we will present a model of expert software design which we have developed in the course of analyzing protocols of expert designers designing an electronic mail system. Two goals motivated this work: The first was to see experts solving problems which called upon their problem solving abilities, as well as their "routine cognitive skills". The second was to create a situation in which general problem solving operators could be seen to interact with a rich knowledge base. Towards these goals we presented expert software designers with a novel and complex problem from a domain with which they were familiar. The result was a model which unites several repeatedly found behaviors into an interpretable whole (Jeffries, Turner, Polson & Atwood, 1981; Kant and Newell, 1982; Atwood, Turner, Ramsey and Hooper, 1979).

2. Methodology

Subjects. Three expert designers. Each of our experts had worked for at least eight years in commercial settings designing a wide variety of software.

Procedure. We presented each of the designers with the following design task to work on.

> ***TASK*** -- Design an electronic mail system around the following primitives: READ, REPLY, SEND, DELETE, SAVE, EDIT, LIST-HEADERS. The goal is to get to the level of pseudocode that could be used by professional programmers to produce a running program. The mail system will run on a very large, fast machine so hardware considerations are not an issue.

The task we gave our subjects had several important properties: It was complex, requiring close to two hours of our expert's time. We hoped therefore to be able to see something of the skills that make up expertise. It was novel, none of our experts had designed a solution to the problem previously. This meant that we would not be seeing only "routine cognitive skill", but some problem solving as well. In addition, the problem we chose was similar to the type of problem with which our experts were familiar. Therefore, although none of our experts had designed a mail system before, they were used to designing other types of communications systems and so we would be able to observe them in a situation where they had rich knowledge bases to turn to.

Organization

In Section 3 we outline the main elements of the model. Next, in Section 4, we present larger portions of our experts' protocols in order to support our claims and to bring out some of the interesting interactions among the components of the model.

[1]This work was sponsored by a grant from ITT.

3. The Model

3.1. Components of the Model

Our model of the experts' design process contains four major elements:

- A Design Meta-Script. The function of the Meta-Script is to drive the design process by setting three general goals:

  1. To check the current state of the design for sufficiency. This means that all of the elements needed to specify the design at the current level are present.
  2. To check the current state of the design for consistency. This means that all elements of the design are compatible at the current level of specificity, and that no element causes an inconsistency with currently known constraints both at higher and at lower levels.
  3. To expand the design from its current level of specificity into the next level of specificity.

  The main operator used to achieve these goals is the running of mental simulations of the partially completed design. The function of these simulations is explained in the next section which describes how the elements of the model function.

- A Sketchy Model. The Sketchy Model resides in working memory and as the design process proceeds the model becomes increasingly less sketchy. The model is initially sketchy in that the designer does not yet understand its functionality down to a level of detail which would be sufficient to produce an implementable program. In addition, the constraints or assumptions of the design are not entirely understood. One way of picturing the model is as a tree that grows in both depth and breadth as the designer's understanding of the problem specification increases (Jeffries et al., 1981; Atwood, Turner, Ramsey, and Hooper, 1979.)

- The Current Long Term Memory Set. This is the set of long term memory elements that are currently under consideration. This set would consist of all of the known solutions appropriate to the aspect of the design that is currently being worked on. Choosing an element from the set currently under consideration allows the designer's model to become less sketchy because the element selected from long term memory is now added to the model in working memory. It also causes a different long term memory set to be considered on the next iteration of the design process.

- The Demons. These contain the designer's notes to himself. The notes are things to remember such as constraints, assumptions, or potential inconsistencies. A note will be placed in a demon if it is: a. too concrete to be resolved at the time that it is thought of and b. needs to be considered when the designer's model has reached a level of concreteness that matches the note. The demons are able to monitor the state of the Sketchy Model. When the level of detail of the Model is equal to the level of detail of the note the demon calls attention to itself. (The reader will notice that our demons are rather unusual in that they are active formation gatherers.)

Summary of the Model

At the most abstract level, the designers were performing a means-ends analysis driven by the Meta-Script. In the analysis the goal state was an implementable design specification and the current state was the designer's increasingly detailed Sketchy Model of the problem solution. The

designers moved towards the goal state by repeatedly simulating executions of their incomplete models. For all of the designers observed, this appears to be the most powerful and frequently used operator in the means-ends analysis. However these simulation runs of the partial models served to decrease the gap between the current and goal states in a number of ways. This will be expanded upon in the next section where we present portions of our protocols. The protocols will help to give the reader a clearer picture of how the model functions. It will also bring out some of the interesting interactions among the components of the model and generate some as yet unaddressed questions.

4. Recurrent Behavior Accounted for by the Model

Here we present behaviors which we found were repeatedly exhibited by our designers. The model unites these behaviors into an interpretable whole.

Observation I. How the Design Proceeded

There was a surprising degree of similarity in the time line of the subjects.

Figure 1: Subjects' Timeline

As illustrated in the figure above, first the designers described how a user would view the mail system, then they expanded upon the various assumptions and constraints of the problem (e.g, S1: "We will assume dumb terminals", S2: "the number of users will not be fixed"). Only then, approximately 20 minutes into the session, did the experts begin to construct a working model of the mail system. This model also changed over time in that it began as a very skeletal version of a mail system and then became increasingly concrete as the design progressed. The following quote from designer S2 illustrates this progression.

At 20 minutes into the task expert S2 said:

> "We can now start thinking about what type of processing structure is required for implementing (the mail system). In order to get the idea about the structure, we can see some kind of a state diagram which shows the dynamics of the system.... What we can see here is one state (accessing) several other states and after the operation is completed, control of the state transition going back to the initial state. This will help us structure our solution to the problem at a higher level. Then we will go into each one of the building blocks that help us write the processing step at each step in the state diagram."

Why did the experts spend the first twenty minutes of the session gathering information? The answer is not that they were hesitant, (e.g. S1's quote ten minutes into the session: "The program design gives me no cause for concern at all.") As discussed in the previous section, the expert

designers search long term memory for stored solution elements. An effective search results from first choosing the right set of memory elements and then choosing the right element from among the set. This type of choice would be aided by having a sufficiently rich set of constraints and assumptions. And it seems that experts do not begin their search until this information has been obtained.

Observation II. Maintaining Balanced Development

The designers always followed a course of *balanced development*. Our subjects attempted to develop each of the components of the design so that none of them acquired significantly more detail than any of the others. The following quote is representative of this behavior. Its significance becomes clear when we consider the next observation, *Simulation Runs of the Sketchy Model.*

> S3: ...you've never got so deep in that you can't improve it taking account of something you think of later... No, oh no I'm not going to take something down to its itty bitty conclusion because bet you I'm going to have to change it. When I take something else out and say (echh??) there's no logical consistency between that, remembering that mail is a two way thing. I'm going to have to reply to it and I may have to use this information to construct a message back ... I can see too many possible interactions between the pieces so it would be nicer if they all had some logical similarity.

Observation III. Simulation Runs of the Sketchy Model.

We observed all of our subjects repeatedly conducting mental simulation runs of their partially completed designs. The experts would consult the state of the sketchy model and then conduct a simulation of the model at its current level of abstraction. Thus we observed simulations which became increasingly concrete as the design progressed. For example, in S3's early simulation he saw the mailer as "information flowing through a system", whereas in a later simulation when considering his module for the READ function, S1 drew a state diagram for all of the states which could be reached from READ. What is the function of these simulations, which appear in our protocols and in those of Kant and Newell (1982)? Recall that the design process is driven by the Meta-Script. The goals of the designer's Meta-Script are to check the current Sketchy Model for consistency and completeness and if these criteria are met the next goal is then to try to expand the Model. Therefore, in order to meet the goals of the Meta-Script a simulation run of the Sketchy Model in its current state is conducted (e.g. S1 draws a state diagram for the current READ module to see how it behaves at this point in its development). We can now see why the designers maintain balanced development; *it would be difficult to run a simulation with elements at different levels of detail.*

Recall that the other goal of the Meta-Script was to expand the Sketchy Model. The simulation is used in this process as well. It points out an element of the Sketchy Model that needs expansion and the subject then accesses the appropriate long term memory set in order to choose how the expansion should proceed. This issue is further addressed below.

Observation IV. Notes and Interrupts

We found that the expert designers would frequently make "notes", to themselves about things to remember later in the design process. These notes had to do with constraints or partial solutions or potential inconsistencies which needed to be handled in order to produce a successful design. The reason that these notes were not handled immediately was that they were concerned with a level of detail which was greater than the level of detail of the current state of the Sketchy Model. This means that incorporating them into the design when they were thought of would

have violated the principle of balanced development. This in turn would have interfered with the process of running simulations, which, as mentioned above, was a process upon which the experts rely quite heavily. We also found that the expert designers would be reminded of previously made notes once the current state had reached a level of detail which would allow the note to be incorporated into the design without violating balanced development. Data of this sort is what led us to posit the existence of demons which were able to monitor the state of the Sketchy Model in order to interact with the design process without disrupting it.

5. Open Questions

Three issues of theoretical significance are raised but not settled by our observations.

- *How do expert designers decide which element in the Sketchy Model to expand upon?* We suggest that, the process of accessing long term memory serves both to guide the articulation of plans for expanding the current node in the design tree and for elaborating goals appropriate to the next level of detail.
- The second stems from the not unlikely possibility that the expert's long term memory will contain several potential solutions for some aspect of a problem. *This raises the question of how the expert chooses among them?* We believe that our expert designers could and did rely heavily on effective communication between the current state of their Sketchy Model and their stored knowledge about software design. That is, we believe that the expert designers' understanding of the current problem served to form an effective index into memory. However, we do not know what this index looks like. We also do not know whether the index handed to memory is the result of heavy inferencing. It could just as well be that memory does the inferencing about what is needed in the current context. Either way, the experts have developed a highly effective retrieval mechanism.
- *How is the Sketchy Model used to perform the simulation?* In our description, the Sketchy Model has the quality more of a structure than of a process, but somehow it is repeatedly used as a running system which the designer can "play with" in order to find emergent properties. The idea of a Sketchy Model needs to be conceived of in a way which takes this aspect of its functioning into account (Collins and Genter, 1982).

6. Concluding Remarks

We have chosen a complex and novel task for our subjects from a domain with which they were familiar. This has allowed us to develop a model which can account for several interesting behaviors such as constraint gathering, balanced development, and the building and running of simulations of partially completed designs. We have proposed a model in which there is a good deal of intelligence given to some of the components (e.g. demons which can understand the current state of the Sketchy Model), as well as frequent interaction between meta-knowledge about design and content specific knowledge about communication systems.

Acknowledgements

Thanks to Ruven Brooks, John Black, Larry Birnbaum, and Kate Ehrlich for their time and thought.

7. References

Atwood, M., Turner, A., Ramsey, R., and Hooper, J. An exploratory study of the cognitive structures underlying the comprehension of software design problems. ARI Technical Report 392. SAI-79-100-DEN. 1979.

Collins, A. and Gentner, D. Constructing Runnable Mental Models. Proceedings of the Fourth Annual Cognitive Science Society. 1982.

Jeffries, R., Turner, A., Polson, P. and Atwood, M. The processes involved in designing software. In Anderson (Ed.) *Cognitive Skills and Their Acquisition*: Hillsdale, NJ: Lawrence Erlbaum Associates, 1981.

Kant, E. and Newell, A. Problem Solving Techniques for the Design of Algorithms. Tech Report No. CMU-C S-82-145 1982.