8-16-1996

DPSS: A CASE Tool for Supporting Design Process of Software Designers

Young-Soo Chung
Department of Management, University of Nebraska- Lincoln

Jong Tae Lee
Department of Computer Science and Engineering, University of Nebraska-Lincoln

DPSS: A CASE Tool for Supporting Design Process of Software Designers

Young-Soo Chung, Department of Management
Jong Tae Lee, Department of Computer Science and Engineering
University of Nebraska-Lincoln

1. Introduction

Wojtkowski and Wojtkowski (1990) classified the research on CASE tools into three broad categories: development of CASE tools, implementation issues, and managerial evaluation of CASE tools impact. Even though CASE technology is increasingly used due to advances in software design methods and hardware technology (Wojtkowski and Wojtkowski 1990), the gains from its adoption have not met expectations. For instance, Jones (1992) argues that CASE tools are lacking a key set of capabilities in management, technical, and support functions that are needed to address chronic problems of software engineering. The research on CASE tools has mainly focused on the implementation issues and managerial evaluation. The focus of this paper is on the relatively neglected area of CASE tools research, the development of CASE tools.

Among three components of CASE tools, upper CASE, middle CASE, and lower CASE, this paper will focus on middle CASE. Middle CASE is a component that supports the systems analysis and design phase (Gibson et al. 1990). The most significant factors of software productivity and quality, such as user requirements satisfaction, delivery speed, reliability, costs, and flexibility, are determined primarily in the analysis and design phase. Hence, the success of CASE tools heavily depends upon how well CASE tools support the design activities.

The primary objective of this paper is to provide a foundation for developing a CASE tool that can adequately support cognitive activities of software designers. The proposed solutions will be presented in a prototype called DPSS (Design Process Support Systems). The rest of the paper is organized as follows. We first identify two areas of cognitive difficulties that individual designers face during design activities: breakdowns due to individual limitations and difficulties in handling the opportunistic problem-solving process. For each area of cognitive difficulties, we suggest possible solutions to reduce designers' mental burden in DPSS. Finally, the conceptual framework of DPSS will be presented.

2. Breakdowns due to individual limitations

During the design process, designers face breakdowns (difficulties) due to lack of design knowledge and cognitive limitations (Curtis et al. 1987; Curtis et al. 1988; Guindon et al. 1987).

2.1 Lack of design knowledge

Design knowledge can be divided into two categories: 1) domain knowledge and 2) knowledge/experience of the design process (Curtis et al. 1987; Curtis et al. 1988; Guindon et al. 1987). When designers lack domain knowledge, they have difficulty understanding the problem itself. Rosson and Alpert (1990) argue that some of the most expensive mistakes in large software development projects can be attributed to the designers' failure to understand the problem situation adequately. When designers lack domain knowledge, they usually do not know the location of resources for understanding and solving the problems. In order to reduce the difficulties due to lack of design knowledge, DPSS will store and provide information about where designers can locate domain knowledge and solutions.

Design schema is a mental representation about a design process. In a design process, previous experience is a major factor that can provide a design schema for the problem solution. For instance, if a designer has experience in solving similar problems, the designer can use the same structure of problem solving process to solve a new problem. In order to understand the problem adequately and to efficiently construct design schema, we believe that it is very important to see how similar problems were resolved and what was the final solution for the expected issues in the early stages of the design. DPSS will store and provide the location of domain knowledge and problem solutions. It will also store and provide the design process in a visualized and arranged format, so that designers can retrieve and learn design schema. For instance, designers could look at the prioritization of issues and selection of alternative solutions in previous problems, thus giving them improved access to domain knowledge.

2.2 Cognitive limitations

During a design process, designers experience cognitive limitations due to 1) capacity limitations of short-term memory and 2) the unreliable retrieval of relevant information from long-term memory (Guindon et al. 1987). In the case of a short design process, the designers may have difficulties in considering all of the stated or inferred constraints in refining a solution. Another difficulty due to cognitive limitation is that the designers find it very difficult to perform complex simulations with many steps or with many test cases.

DPSS will reduce designers' mental burden by presenting all of the stated or inferred constraints in an arranged format. It will also help designers to perform complex mental simulations by showing sequential steps or many test cases in order. In particular, since the designers can see the process through visual symbols, it will enable the designers to visualize the solutions.

3. Support for opportunistic problem solving

A fresh view on software design process is that software development is a problem solving process (Curtis et al. 1987; Curtis et al. 1988; Rosson and Alpert 1990). As Simon (1973) noted, design is an ill-structured problem involving complex cognitive activities. Currently, the vast majority of CASE tools only support to document solutions, but do not provide designers genuine tools for deriving solutions.

In idealized software design, the design process is top-down in which problem decomposition is breadth first, with goals identified at one level are then broken down into

more and more detailed levels (Rosson and Alpert 1990). Currently, CASE tools are designed to support only top-down decomposition of problem solving. However, Guindon (1990) argues that the top-down decomposition is problematic in the early stages of the design. Instead, he suggests that an opportunistic decomposition is better suited to handling the ill-structuredness of design problems and that top-down decomposition is suited only for well-structured problems in which designers already know the correct decomposition. Opportunistic design is a design approach in which interim decisions can lead to subsequent decisions at various levels of abstraction in the solution decomposition. According to Guindon (1990), the important causes of opportunistic design are the sudden additions and inferences of new requirements, partial solutions triggered by data-driven rules and associations, and drifting through partial solutions.

As the software development process tends to be more complex, large, and integrated, the design process usually involves a long period of time. It becomes very difficult for designers to remember all problems and subproblems, and the essential relationships among them. In order to support designers' opportunistic problem solving, DPSS will allow designers to expand a problem in any direction, either top-down or bottom-up. It will also allow designers to store postponed or partial solutions, so that the designer can resume the design process without difficulty when problems are resolved.

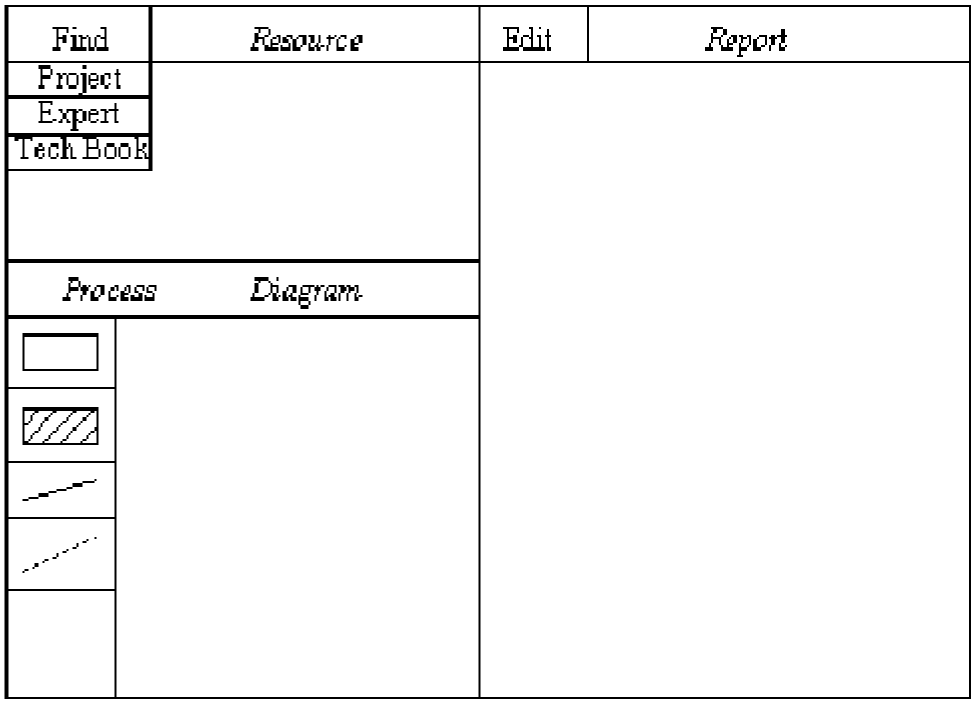
4. Description of DPSS

A conceptual framework of DPSS is presented in <Figure 1>. In <Figure 1>, there are three major windows: Resources, Process Diagram, and Report. The Resources window, which is upper left, provides functions related to finding resources for a problem. Resources can be found by a keyword to find appropriate projects, experts, and technology books. The Process Diagram window, shown at lower left, provides functions for representing the problem-solving process through visual symbols. This window helps to visualize the problem-solving process, so that the designers can look at the constructed diagram and simulate the problem-solving process. By using different types of symbols, the designers can indicate the problem or issue as a final or postponed solution. In order to support opportunistic problem solving, the diagram can start at any level of abstraction. Finally, the Report window, at the right part of the window, provides functions that enable designers to edit reports. Each symbol in the Process Diagram window represents a report that can be edited in the Report window. The report consists of a problem description, related high-level and low-level issues, alternative solutions, and the location of resources. The contents of this report are not focused on the documentation of final solutions as found in ordinary CASE tools, but are more for the documentation of a particular problem-solving process.

5. Future work

In the previous sections, we have identified cognitive difficulties that designers face during design activities and have suggested possible solutions for them in DPSS. We believe that the only possible way to verify our framework is actually to build a prototype and look and feel the functions. We are currently implementing DPSS with HyperCard on Macintosh. We are also preparing scenarios of realistic cases to explain and verify the functions of DPSS.

Figure 1. DPSS screen



References

Curtis, B., Krasner, H., Shen, V., and Iscoe, N. "On Building Software Process Models Under the Lamppost," *Proceedings of the 9th International Conference on Software Engineering*, 1987, pp. 96-103.

Curtis, B., Krasner, H., and Iscoe, N. "A Field Study of the Software Design Process for Large Systems," *Communications of the ACM*, November 1988, pp. 1268-1287.

Gibson, M.L., Snyder, C.A., and Carr, H.H. "Implementing a Corporatewide Information Strategy Through CASE," *Journal of Information Systems Management*, Summer 1990, pp. 8-17.

Guindon, R., Krasner, H., and Curtis, B. "Breakdowns and Processes During the Early Activities of Software Design by Professionals," in *Empirical Studies of Programmers*, Vol. 2, G. Olson, E. Soloway, and S. Sheppard (eds.), Albex, Norwood, NJ, pp. 65-82, 1987.

Guindon, R. "Designing the Design Process: Exploiting Opportunistic Thoughts," *Human-Computer Interaction*, Vol. 5, 1990, pp. 305-344.

Jones, C. "CASE's Missing Elements," *IEEE Spectrum*, June 1992, pp. 38-41.

Rosson, M.B. and Alpert, S.R. "The Cognitive Consequences of Objective-Oriented Design," *Human-Computer Interaction*, Vol. 5, 1990, pp. 345-379.

Simon, H.A. "The Structure of the ill structured problems," *Artificial Intelligence*, Vol. 4, 1973, pp. 145-180.

Wojtkowski, W. and Wojtkowski, W.G. "A Look at the Status of CASE Technology," in *Research Issues in Information Systems: An Agenda for the 1990s*, A. Jenkins, G. Wojthawkin, and W. Wojtkowski (eds.), Wm. C. Brown, Dubuque, IA, 1990, pp. 173-191.