
ABSTRACT: While the proliferation of computer-aided systems engineering (CASE) tools in the professional world mandates their exposure to information systems (IS) students, many IS faculty may be reluctant to introduce CASE due to the seeming paradox of learning a CASE tool before acquiring a thorough knowledge of a systems development methodology. Unfortunately, an unproven tenet of CASE implementation is that a thorough knowledge of a systems development methodology is necessary before attempting to use CASE. By adhering to this tenet we may be precluding students from using CASE technology in their information systems curricula. Researchers in the field of education, however, view technology as a useful tool in learning. Studies in computer-aided instruction indicate that the feedback provided by software can be especially useful during the learning process. By applying what is known about computer feedback to CASE tools it may be possible to use CASE as a tool for learning a systems development methodology as well as producing quality systems.

INTRODUCTION

While the proliferation of computer-aided systems engineering (CASE) tools in the professional world [1] mandates exposure of the tools to information systems (IS) students, many IS faculty may be reluctant to introduce CASE due to the seeming paradox of learning a CASE tool before acquiring a thorough knowledge of a systems development methodology. This is in spite of the fact that an important role of a CASE tool is to serve as a methodology companion, i.e., to assist in the creation of documentation passed to succeeding phases of the systems development life cycle and to guide the user through a particular systems development methodology [2]. Although CASE tools are frequently designed to support a particular systems development methodology, a frequent stumbling block in the CASE implementation process (both in the professional world and in academia) has been the potential conflict between the goals of learning a systems development methodology and learning a CASE tool. The CASE implementation literature prescribes a thorough knowledge of the chosen systems development methodology as a prerequisite to learning and using CASE [3, 4, 5, 6]. Despite the fact that this "rule of thumb" is based solely on anecdotal evidence it appears to have been accepted as a tenet of CASE implementation.

While the view of the information systems community has been to learn the systems development methodology before attempting to use CASE technology, a different philosophy regarding the role of technology in the learning process is adhered to by many researchers in the field of education. The literature in the area of computer-aided instruction (CAI) indicates that technology can be effectively used as a mechanism for learning. In support of using the computer as an agent for instruction Ledgeld [7] states that the computer can be used to rapidly diagnose learner errors, provide feedback to keep a learner focused, and enhance experiential opportunities. Lipson and Fisher [8] provide the following description of the use of a word processor to teach and promote writing skills:

"Computer-based word processors are specialized tools that greatly simplify the mechanics of writing, editing, correcting, and retyping a manuscript. They facilitate manuscript production by minimizing the mechanical details of writing and permitting maximum attention to the flow of ideas. With word processors students can achieve a higher level of quality in their creative and technical writing. With the mechanical details of writing minimized, students can pay greater attention to content, form, and style. And with the satisfaction that comes from producing clean, professional looking copy, students may be more motivated than ever before to tackle the task of composition (p. 254)."

The use of a word processor as a tool to assist in the development of writing skills parallels the potential use of CASE as a pedagogical instrument for learning a systems development methodology. Similar to the word processor simplifying the learning of the writing process, as well as the mechanics of writing, CASE, through its role as a methodology companion, may be used to simplify learning and using a systems development methodology. Rather than viewing a CASE tool as simply a means for improving system quality or enhancing productivity, this paper proposes using CASE as an aid for learning a systems development methodology. Specifically, CASE is examined with respect to the potential feedback it can give a student learning a systems development methodology. First, the concept of feedback in computer-aided instruction is reviewed, as are studies evaluating the impact of feedback on CAI. From this brief review of the literature, four components are identified that serve to describe the type of feedback provided by a CAI system. Next, the four components of feedback are
There are two types of methodology feedback that can be presented to a student by a CASE tool: restrictive feedback and guided feedback.

The study shows that this feedback is especially helpful to those students who are not well-versed in the subject matter.

In summary, four components of feedback have been identified, which, taken together, define the feedback offered by a CAI system to a student: 1) immediacy, which indicates when the feedback is provided; 2) solicitation, which indicates how the student receives the feedback; 3) content, which indicates why the feedback is being provided; and 4) response, which indicates what the required reaction to the feedback must be. In the following section, the components of feedback are examined within the context of CASE technology.

METHODOLOGY FEEDBACK MECHANISMS IN CASE

In order to provide methodology feedback to a student, methodology rules must be embedded within a CASE tool in such a way as to allow the CASE tool to notify the student of any mistakes, inconsistencies, or instances of incompleteness encountered while using the tool. There are two types of methodology feedback that can be presented to a student by a CASE tool: restrictive feedback and guided feedback. Restrictive methodology feedback can be defined as any feedback that requires a student to adhere to the rules of a chosen methodology. In order to determine if the methodology feedback is restrictive, the four components of feedback identified in the previous section must be examined within the

---

**Figure 1: Restrictive Feedback**

<table>
<thead>
<tr>
<th>Rule Violation</th>
<th>When?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immediate</td>
<td>Level 1</td>
</tr>
<tr>
<td>How?</td>
<td>Post-Method</td>
</tr>
<tr>
<td>Solicitation</td>
<td>Request</td>
</tr>
<tr>
<td>Why?</td>
<td>Simple Notification</td>
</tr>
<tr>
<td>Content</td>
<td>Override</td>
</tr>
</tbody>
</table>

JOURNAL OF INFORMATION SYSTEMS EDUCATION Fall 1995
context of providing CASE methodology support (see Figure 1). The immediacy of the feedback refers to when a rule violation is presented to the student. Methodology feedback that is restrictive will be presented as soon as is feasible to do so in order to keep any methodology violation from propagating through the system specification. This implies that rule violations must be detectable while in the process of performing a specific task, such as data flow diagramming (Level 1 Restriction). Other rule violations may not be detectable until the student is finished with a task (e.g., while saving a DFD and/or exiting the diagramming tool). This second level of restrictive feedback (Level 2 Restriction) is necessary in order to prevent a student from being interrupted by mistaken violations that are actually attributable to work in progress. The solicitation of a rule refers to the mechanism by which the rule violation is presented to the student. Feedback that is implemented in a restrictive fashion by a CASE tool will automatically present itself to the student as soon as a violation is detected by the CASE tool. With the goal of restrictive feedback being to force the student to conform to the rules of a particular methodology, it is important that the content of the feedback be specific to the rule violation. The response refers to the set of options available to the student once the feedback has been presented by the CASE tool. Restrictive feedback will require the student to address the feedback, i.e., correct the violation, before proceeding further. In summary, rule feedback will be considered to be implemented within the CASE tool in a restrictive fashion if the student is automatically presented with descriptive feedback while using an operator, or while terminating use of an operator, and is forced to address the feedback before proceeding.

The second type of methodology feedback available to a student from a CASE tool is guided feedback. Guided feedback can be defined as any feedback that guides a student in executing a systems development methodology by assisting the student in choosing and using its methods. A CASE tool’s feedback may guide the execution of the systems development process by providing the student with suggestions and information regarding the procedures of a particular systems development activity as well as the resultant product of that activity. Two types of guided feedback can be made available to a student by a CASE tool: active guidance and passive guidance. Active guidance is informative and suggestive advice that is unsolicited, i.e., the CASE tool delivers the feedback to the student when the CASE tool detects a need for guidance (see Figure 2). Active guidance can be provided by the CASE tool while the student is performing a particular task (Level 1 Active Guidance) or it may be provided by the CASE tool when the student is finished with the task. For example, while saving and/or exiting (Level 2 Active Guidance). The feedback may be presented to the student in the form of an error notification and/or suggestions for correcting the violation. It is then left to the discretion of the student to determine whether or not to correct the violation.

The second type of guided feedback provided by a CASE tool is passive guidance. Passive guidance is informative and suggestive advice that is solicited by the student from the CASE tool (see Figure 3). Passive guidance may be requested by the student while performing a task (Level 1 Passive Guidance) or it may be implemented as a separate function outside of the task (Level 2 Passive Guidance). As with active guidance, feedback may be presented to the student in the form of error notifications and/or suggestions for correcting the violations.

Finally, an alternative to embedding restrictive and guided feedback within a CASE tool is the complete lack of feedback support for a methodology or a particular methodology rule.

Research Directions
The previous section explored the range of methodology feedback that a CASE tool can provide a student. Unfortunately, empirical work investigating the impact of CASE feedback on either learning a systems development methodology or developing quality specifications has been limited.

To determine if the use of CASE has a negative impact on learning introductory systems development concepts, including methodologies, Heiat and Heiat [17] conducted two introductory systems analysis and design courses, each of which used a different approach toward teaching a systems development methodology. During the first course no CASE tools were used, while in the second course the students received hands-on training in CASE while they were learning a systems development methodology. In both classes the students were twice examined on their understanding of the course principles. No significant difference in performance was found between the two groups of students, suggesting that the use of CASE does not have adverse effects on student learning. Because the CASE tool chosen for this study, BriefCASE, offers virtually no methodology feedback to the student, the opportunity exists to repeat this study with CASE tools offering various levels of methodology support.

The feasibility of applying feedback to the rules of structured analysis is examined by Jankowski [18]. This study reveals that the rules of the structured analysis methodology cannot all be supported by the same type of feedback, i.e., for some rules restrictive feed-
Case Feedback

continued from page 99

back is possible while for other rules only guided feedback is possible. The framework presented can be considered an upper bound on feedback (i.e., most restrictive) and can be used as a benchmark for comparing CASE tools for academic adoption.

A second study by Jankowski [19] examined two CASE tools that are frequently utilized to support information systems course work: Intersolv’s Excelerator 1.9 and Visible Systems’ Visible Analyst Workbench 3.1. Each CASE tool was used by eight student project teams to develop a functional specification for a hotel information system. For each structured analysis rule involving data flow diagramming, minispecs, and the data dictionary, the number of rule violations in the system specification were recorded. The results indicate that the level of feedback provided by the CASE tool does not impact the rules applying to a particular data flow diagram (e.g., a process must not be free-standing). However, for the rules that apply to the parent-child relationships between the diagrams, and the relationships between the diagrams, the data dictionary, and the minispecs, rule violations were recorded less frequently when restrictive feedback was supplied by the CASE tool than when passive guidance feedback or no feedback was supplied.

CONCLUSION

Future work in this area may reveal that CASE, when providing the proper feedback, may be an appropriate tool for students and professional analysts learning a systems development methodology. Further, the results might also point the way toward the establishment of CASE tools that offer variable feedback that is dependent upon the experience of the user. Based upon previous research in the area of CAI, and encouraging preliminary results in MIS research, it may soon be possible to improve the notion that a systems development methodology must be thoroughly understood before attempting to support it with a CASE tool.

REFERENCES


David John Jankowski, Ph.D.
Center for High Technology Management
College of Business Administration
California State University, San Marcos
San Marcos, CA 92096-0001
(619) 752-4235
david.jankowski@csusm.edu

David John Jankowski is an assistant professor of management information systems at California State University, San Marcos. He received his Ph.D. from the University of Arizona. Prior to his graduate studies he spent three years as a software engineer for a Southern California defense contractor. His research interests include CASE, group tools for systems analysis and design, and the information superhighway.

Future Software

continued from page 105

examine the shifting needs of both business and academia.

Further, expanding this research region by region until all fifty states have been surveyed, and comparing the findings would make the results more inclusive and reliable.

REFERENCES

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.