Journal of Universal Computer Science, vol. 1, no. 3 (1995), 195-200
submitted: 1/3/95, accepted: 13/3/95, appeared: 28/3/95, ©Springer Pub. Co.

Bringing ITS to the Marketplace: A Successful Experiment in Minimalist Design

Carl Gutwin and Marlene Jones
Alberta Research Council, Calgary, Canada
E-Mail: marlene@arc.ab.ca

Patrick Brackett and Kim Massie Adolphe
Gemini Learning Systems Inc., Calgary, Canada
E-Mail: kadolphe@gemini.com

Abstract: Intelligent Tutoring Systems (ITS) have proven to be effective tools for teaching and training. However, ITSs have not become common in industrial and organisational settings, in part because their complexity has proven difficult to manage outside of the research lab. Minimalist ITSs are an attempt to bridge the gap between research and practical application; they simplify research techniques while striving to maintain as much pedagogic intelligence as possible. This paper describes one such system, SWIFT, that is an example of how a minimalist ITS can be delivered as a commercial product. We outline some of the issues facing designers of a minimalist system, and describe the ways that research techniques have been incorporated into four modules of SWIFT: adaptive testing, course planning, guidance, and diagnosis.

1 Introduction

Despite their many successes [see Shute, 1990], intelligent tutoring systems are underemployed in the world's industries and organisations. Complexity, size, and lack of tools to assist course authors have all prevented systems developed in research labs from moving to the commercial world. These limitations point to the need for techniques and strategies that can adapt research results to industrial and commercial systems. The idea of minimalist ITS (e.g. [Brusilovsky et al., 1994], [Winne et al., 1992]) has evolved to fill that need, and SWIFT is an example of a successful attempt to bring a minimalist ITS to the marketplace.

SWIFT, shown in Figure 1, is an adaptive learning environment [Jones, 1992] that has been developed in a joint research venture between the Alberta Research Council and Gemini Learning Systems. Several of SWIFT's

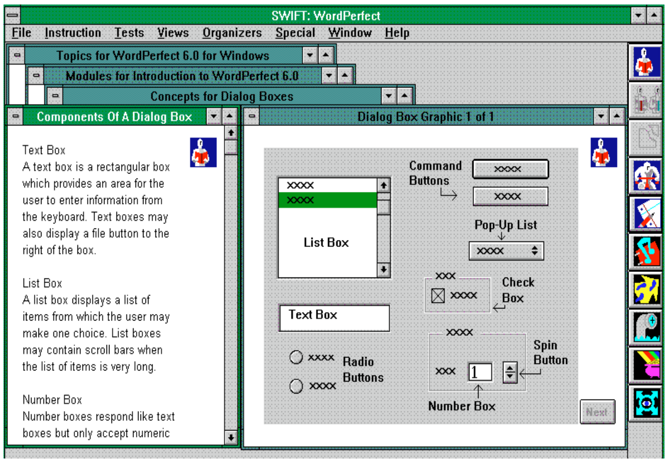

Figure 1. The SWIFT interface.

elements incorporate techniques from ITS research, including an adaptive testing facility, a situation guide, a course planner, and a diagnosis module. SWIFT shows that innovative techniques from research can be successfully modified for use in the real world. The following sections examine more closely what it means to minimise an ITS, and then discuss the design of SWIFT as a minimal system.

2 Minimalist ITS

Many intelligent tutoring systems require computational power that is rare in real-world training situations. The desire to produce training systems that behave intelligently, but that are also feasible on a smaller scale, has led researchers to the idea of minimalist ITS. This area has grown from work on training shells (e.g. [Major & Reichgelt, 1992], [van Marcke, 1990]) and on discovery learning environments (e.g. [Shute & Glaser, 1990], [Elsom-Cook, 1990]). Minimalist ITSs attempt to bridge the gap between research and practical application by simplifying ITS techniques for use in computing environments with limited memory, external storage, and processing speed. Any of the common elements of a typical ITS, such as the domain representation, the learner model, the pedagogic and domain expertise, the instructional and delivery planners, or the diagnosis engines, can be minimised. In addition, minimalist systems often attempt to reduce demands on course designers, since teachers and industrial trainers often do not have the time nor the specialised knowledge necessary to build the knowledge structures used by existing ITSs.

Meeting these needs presents a minimalist designer with several tradeoffs, the most obvious of which is the balance between power and feasibility. To manage this tradeoff effectively, a designer must understand what is lost with each simplification of a technique, and must look for other ways to bolster the system's pedagogic capabilities. In most cases, the minimalist approach implies that a system will gather less information about the student, will have a less-sophisticated domain representation, and will be able to make only relatively straightforward inferences from that information. Additional means must be found to ensure that what is left of the ITS techniques can still be used to advantage, which often means combining the technique with other more robust mechanisms that can make up for reduced intelligence. Our experiences suggest that the minimalist goal involves more than just linear scaling of the techniques–a simplistic approach will result only in a bad approximation of an ITS, not a minimalist one.

One approach that we have taken in SWIFT is to involve the learner in the decision-making processes. Even novices have considerable knowledge about their own learning needs, and they can monitor and alter the system's decisions if given the appropriate support. Other minimalist approaches take strategies from traditional computer-based training (CBT) and enhance them with ITS techniques. For example, CBT is often based upon learning by presentation and questioning. By providing individualised feedback on a learner's answers, an instructional system can add the individualisation of ITS to the simplicity of CBT.

Course designers also play an important part in the successful application of minimalist ITS. Authoring must be made accessible so that domain experts can readily construct new training materials; they must also be able to understand the strengths of the tutoring system in order to maximise its capabilities. KAFITS [Murray & Woolf, 1992] and COCA [Major, 1994] are two examples of systems that concentrate on providing tools for authors; SWIFT also provides extensive support for course design [Massie, 1994].

3 Minimalist Design in SWIFT

The following sections describe the approaches that we have used to make the most of the resources available to our system. In general, our strategies have taken three paths: first, we have found ways to minimise ITS techniques without compromising too much of their power; second, we have found additional mechanisms to make our solutions more robust; and third, we have taken advantage of the abilities of learners and our knowledge of the eventual user population.

3.1 Knowledge Representation

A defining feature of an ITS is a semantic representation of the instructional domain, where concepts are encoded in data structures that allow the system to reason about the course. A minimalist ITS must also employ semantic representation, for an understanding of the concepts in the domain is the basis of much of a system's intelligent behaviour. However, the detail and sophistication of the representation can vary. In SWIFT, we have implemented a representation scheme that allows us to reason about the domain, but does not contain as much detail about specific concepts as might be found in a full ITS. SWIFT courses are stored in a hierarchical structure that divides the instructional material into smaller and smaller pieces, much as a book does with chapters, sections, and subsections. A course has three levels: the first contains a set of *topics*, which are divided at the second level into sets of *modules*, which are divided at the third level into *concepts*. A semantic representation of the course also allows the specification of dependencies between concepts. The current version of SWIFT allows for prerequisite and sequence links between individual concept objects.

3.2 Adaptive Testing

ITSs gather information about a learner's progress by observing them as they interact with the learning environment. Many minimalist systems use exercises, quizzes, and exams as the setting for these observations, since the range of possible inference about the learner can be more easily constrained. Since many organisations (corporate and otherwise) also require that a training system provide concrete records of progress, we have chosen to use formative and summative testing as our means for observing the learner in SWIFT.

One of the problems with traditional exams is that they are of fixed length; a learner must complete a long series of questions in order for the system to determine how well they know a subject. This characteristic can cause frustration for both novices and experts, who may know after a few questions that the subject matter is either bewildering or trivial. Aside from giving the learner greater control over exams–in that they are never forced to take a test–our primary strategy for tackling the problem of fixed-length exams is adaptive testing. Adaptive testing allows exams to be significantly shorter than traditional tests, without losing any predictive power about a learner's mastery of the material. The approach that is implemented in SWIFT is based on the work of [Welch & Frick, 1993]. The algorithm uses Bayes' theorem to estimate the probability that the learner is a master or non-master of the material after each test question is answered. In SWIFT, novices (non-masters) and experts (masters) can be determined in as few as five questions.

3.3 Instructional Planning

Instructional planning in SWIFT is based on two information sources: the results of an adaptive pretest, and the learner's own choice of one or more instructional goals. Each goal specifies which topics and modules of the course are to be included in the learner's path; performance on the pretest then indicates whether concepts within those sections are already known and need not be included. Our approach to instructional planning is effective, but is relatively simple compared to some ITSs (e.g. [Brecht, 1990]) because of SWIFT's less-sophisticated domain representation. Since our simpler approach weakens SWIFT's planning to a degree, we have found other ways of ensuring that appropriate instruction is always available to the learner.

Since we knew that the target population for SWIFT is composed largely of learners that are cooperative and motivated, we were able to view instructional planning as a human-computer problem rather than just a computational one. One of the ways we involve the learner is by providing tools that allow them to monitor their path through the course, and to take control if desired. Figure 2 shows a concept map in SWIFT, one of several displays that explicitly lay out the content and dependencies of a course, and allow the learner to make informed decisions about what to learn next. This approach improves instructional planning by making use of the

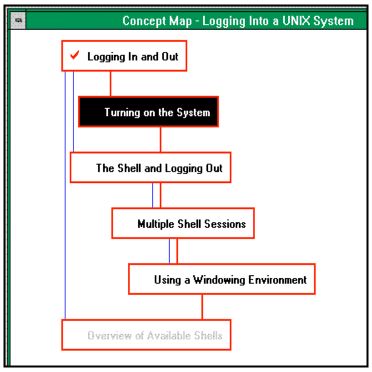

Figure 2. A SWIFT Concept Map.

knowledge of both parties: learners can improve upon or customise the system's course plan if they wish; the recommended path, which is adequate in most cases, provides support for learners who do not wish to venture out on their own.

3.4 Diagnosis

Diagnosis modules attempt to understand problems and misconceptions in a student's knowledge of the domain (e.g. [Johnson & Soloway, 1985], [McCalla & Greer, 1990]). Although any student action may be considered, diagnosis is commonly applied to a learner's answers to test or exercise questions. Diagnosis entails drawing conclusions about the learner's knowledge based on features in their answers; good diagnosis allows systems to provide appropriate feedback and remediation as well as simple indications of whether an answer is right or wrong.

Diagnosis can require significant inferencing power and domain knowledge, which are not the strengths of minimalist systems. An alternative to a fully knowledge-based approach is to detail a number of categories, or cases, of typical errors and misconceptions. Using a case-based approach transforms the inference problem to one of classification, but effective classification can also be difficult to achieve. One problem occurs in specifying the answers that belong to a particular class. The obvious method is to encode every answer. However, this technique implies that any variation of an answer, even those that do not change its essential parts, must also be included. This can be a daunting task for any but the most trivial of exercises.

Our approach to this problem allows a course designer to concentrate on the qualitative differences in the possible answers to a question, rather than on syntactic variations. Our case-based diagnosis subsystem uses regular expressions, constructs that allow a designer to specify a large number of possible variations with a single answer pattern. The system can examine and evaluate any short textual answers for which cases have been designed. The course designer specifies patterns for classes of correct and incorrect answers, and can annotate each class with appropriate feedback and remediation information.

This strategy still requires that the course designer understand the kinds of difficulties that learners can have in a particular area, and how each problem can be manifested in answers to questions. However, we have provided a framework for structuring and using that pedagogic knowledge that is both powerful and efficient enough to be used in a minimal system.

3.5 Situation Recognition and Guidance

SWIFT has more and more become a learner-controlled system, both by design and by necessity. In a self-directed environment, the task of the intelligent tutoring system shifts from tutoring and control to guidance and support. We have been forced to find and implement mechanisms for supporting learners as they explore the system on their own.

We have developed a subsystem within SWIFT that can provide guidance on pedagogic issues according to the specific situation that the learner is in, and can also encourage the learner to initiate certain learning behaviours. Many strategies exist for assisting self-directed learning that promote metacognition and more effective learning behaviour (e.g. [Derry & Murphy, 1986], [Derry, 1992], [Pressley et al., 1989], [Shuell, 1992], [Winne, 1992]). Examples of effective learning behaviour include positive self-talk, note-taking or highlighting, summation, imagery, question-generation, and review of learning objectives.

SWIFT's guide watches system events and monitors a learner's location, history, and current knowledge. When particular kinds of situations occur, the guide can decide to deliver advice to the learner. For example, if a student turned their attention to a new section of course material, the guide might suggest that they test their knowledge of the current section before going on. The guide is implemented as a rule-based system, and the above example would involve a rule such as: "if the learner has not demonstrated mastery in the concepts of the current module, and the learner requests a move to a new module, the system will suggest that the learner take a module test for the current module." The guide's advice is presented in a popup dialogue box, such as the one shown in Figure 3.

The rule-based guidance system provides SWIFT with a generalised architecture for presenting useful information. We are able to give the learner pedagogic guidance in a wide variety of situations, but the

Figure 3: A SWIFT Guidance Window

architecture can also be used to give information about any situation, such as tips on using SWIFT to its full capacity.

4 Current Success and Plans for Further Work

SWIFT has now been released as a commercial product, and a number of organisations are producing courses in domains as varied as air traffic control, Canadian history, high school physics, and football. Designers have so far found course development to be straightforward, but we are planning for a graphical authoring environment to further support the authoring process. SWIFT has been evaluated through formal usability studies involving representative users from industrial settings and from secondary and post-secondary education institutions. The usability testing has validated many of our design decisions, but has also caused us to refine some parts of the system. For example, some users felt that the guide offered advice in too many situations; we have since tuned the guide's rules to reduce repetitive or spurious advice.

We are currently investigating other techniques from ITS research that may be appropriate for implementation in a minimalist system, as well as planning commercial improvements such as more sophisticated multimedia and hypertext support. Some of the possibilities for the next version of SWIFT are:

- Granularity-based diagnosis [McCalla and Greer, 1994];
- Collaborative learning tools, such as support for awareness of other learners (e.g. [Ayala and Yano, 1994]);
- Improved student modeling based on recent techniques of modeling learners based on test results [Shute, 1994].

Our experiences have shown us that minimalist thinking involves more than taking existing techniques and scaling them down, and that additional mechanisms must often be found to ensure robust and rewarding interaction with a minimalist ITS. The examples of our design efforts in SWIFT suggest that minimalist ITS can be successfully constructed within the constraints of the commercial world.

SWIFT is available from Gemini Learning Systems, Inc. and from its distributors.

5 References

[Ayala & Yano, 1994] Ayala, G. and Yano, Y. (1994) Design Issues in a Collaborative Learning Environment for Japanese Language Patterns. *Educational Multimedia and Hypermedia*, AACE, 67-72.

[Brecht, 1990] Brecht (Wasson), B. (1990) Determining the Focus of Instruction: Content Planning for Intelligent Tutoring Systems. Unpublished doctoral dissertation, University of Saskatchewan.

[Brusilovsky et al., 1994] Brusilovsky, P., van Marcke, K., Murray, T., Major, N. and Vassileva, J. (1994) Minimalist ITS. Panel Discussion, *Educational Multimedia and Hypermedia*, AACE.

[Derry & Murphy, 1986] Derry, S., and Murphy, D.A., (1986), Designing Systems that Train Learning Ability: From Theory to Practice. *Review of Educational Research*, 56(1), 1-39.

[Derry, 1992] Derry, S. (1992) Metacognitive Models of Learning and Instructional System Design. In *Adaptive Learning Environments: Foundations and Frontiers*, M. Jones and P. Winne ed. Springer-Verlag. 257-286.

[Elsom-Cook, 1990] Elsom-Cook, M. (1990) *Guided Discovery Tutoring*. London: Paul Chapman Publishing.

[Johnson & Soloway, 1985] Johnson, W. L. and Soloway, E. (1985) PROUST: An Automatic Debugger for Pascal Programs. *Byte*, 10 (4), 179-190.

[Jones, 1992] Jones, M. (1992) Introduction. In *Adaptive Learning Environments: Foundations and Frontiers*, M. Jones and P. Winne ed., Springer-Verlag, 1-10.

[Major, 1994] Major, N. (1994) Evaluating COCA - What do teachers think? *Educational Multimedia and Hypermedia*, AACE, 361-366.
[Major & Reichgelt, 1992] Major, N. and Reichgelt, H. (1992) COCA: A Shell for Intelligent Tutoring Systems. *Intelligent Tutoring Systems*, Springer-Verlag, 523-530.
[Massie, 1994] Massie Adolphe, K. (1994) AI and Education. *Canadian AI Magazine*, 34 (1), 4-8.
[McCalla & Greer, 1990] McCalla, G. I. and Greer, J. E. (1990) SCENT-3: An architecture for intelligent advising in problem-solving domains. In *Intelligent Tutoring Systems: At the Crossroads of Artificial Intelligence and Education*, C. Frasson and G. Gauthier eds. Norwood, NJ:Ablex, 140-161
[McCalla & Greer 1994] McCalla, G.I, and Greer, J.E. (1994), Granularity-Based Reasoning and Belief Revision in Student Models, In *Student Modelling: The Key to Individualized Knowledge-Based Instruction*, J. Greer and G. McCalla eds, Springer-Verlag, 39-62.
[Murray & Woolf, 1992] Murray, T. and Woolf, B. (1992) Tools for Teacher Participation in ITS Design. *Intelligent Tutoring Systems*, Springer-Verlag, 593-600.
[Pressley et al., 1989] Pressley, M., Johnson, C., Symons, S., McGoldrick, J., Kurita, J., (1989), Strategies That Improve Children's Memory and Comprehension of Text. *The Elementary School Journal*, 90(1), 3-32.
[Shuell, 1992] Shuell, T. J. (1992) Designing Instructional Computing Systems for Meaningful Learning. In *Adaptive Learning Environments: Foundations and Frontiers*, M. Jones and P. Winne ed., Berlin:Springer-Verlag, 19-54.
[Shute, 1990] Shute, V. (1990) Rose Garden Promises of Intelligent Tutoring Systems: Blossom or Thorn?, Space Operations, Applications and Research (SOAR) Symposium, June 1990, Albuquerque, NM.
[Shute, 1994] Shute, V. (1994) Regarding the I in ITS: Student Modelling. In *Educational Multimedia and Hypermedia*, AACE, 50-57.
[Shute & Glaser 1994] Shute, V. J. and Glaser, R. (1990) A Large-scale Evaluation of an Intelligent Discovery World: Smithtown. *Interactive Learning Environments*, 1(1), 51-77.
[van Marcke, 1990] van Marcke, K. (1990) "A Generic Tutoring Environment." *The European Conference on Artifical Intelligence*, 655-660.
[Welch & Frick 1993] Welch, R.E., and Frick, T.W. (1993), Computerized-Adaptive Testing in Instructional Settings. *Educational Technology Research and Development*, 41(3), 47-62.
[Winne, 1992] Winne, P. (1992) State-of-the-Art Instructional Computing Systems that Afford Instruction and Bootstrap Research. In *Adaptive Learning Environments: Foundations and Frontiers*, M. Jones and P. Winne ed., Berlin:Springer-Verlag, 349-380.
[Winne et al., 1992] Winne, P., Butler, D., McGinn, M., Sugarman, J., Jones, M., Mark, M., and Field, D. (1992) STUDY: A Tool for Authoring Adaptive Learning Environments and for Advancing Instructional Research. In Appendix to *Proceedings of ITS'92*.

Acknowledgments

Several people have contributed to the design of SWIFT and played a part in the ideas presented here. Thanks to Stuart Williams, Ruby Loo, Joseph Poon, Julia Driver, and Jim Tubman. Special thanks to Jim Tubman and Pam Hirtle for assistance in long-distance preparation of the final draft.