

How to Cut CBI Development Costs in Half: Commentary on Foshay and Preese

JOSEPH M. SCANDURA*

*MERGE Research Institute
Visiting Research Professor, Drexel University
Adjunct and Emeritus Professor, University of Pennsylvania*

Cost estimates presented in Foshay and Preese are based on a long commercially successful history and, hence, are particularly valuable in comparing alternative authoring methods. AuthorIT provides an extreme case because it automates many authoring processes: a) The need to program pedagogy (delivery modes) is completely eliminated, irrespective of the structure of the content to be delivered. b) Dialog-box-based configurability enables AuthorIT to support multiple pedagogies -- ranging from self-directed to highly adaptive -- without programming and at no additional cost. Both are of critical importance in maintaining quality control and reducing costs. While further work is planned, AuthorIT already supports most kinds of learning and tutorial development. Prototype development to date has yielded time and cost savings of 50% or more.

Keywords: Authoring system, computer based instruction, adaptive tutoring, configurable tutoring, structural learning theory, abstract syntax trees, AuthorIT, AutoBuilder, TutorIT.

Development costs have long imposed practical constraints on developing quality CBI. The cost estimates presented in Foshay and Preese are particularly valuable because they are based on a long commercially successful history of development in the field.

*Corresponding author: scandura@scandura.com; www.scandura.com

Role of AuthorIT in Problem Solving.— Accordingly, it is of some interest to compare this base line against prospective alternatives. In this context, AuthorIT (Scandura, 2005) provides an extreme case. AuthorIT includes a component, called **AutoBuilder**, which supports the construction of SLT rules in which both structural (declarative) and procedural knowledge are represented as ASTs. Another, called **Blackboard Editor**, is used to define the interface through which TutorIT interacts with learners. A third **Configuration Tool** makes it possible for the author to specify how TutorIT is to deliver instruction to the learner. **TutorIT** includes content-independent mechanisms for pinpointing what individual learners do and do not know, and what they need to succeed. It also supports teaching declarative as well as procedural knowledge as well as transitions from procedural to expert knowledge.

As detailed in Scandura (2005) ASTs make it possible to represent knowledge simultaneously at all levels of abstraction. ASTs, as defined in SLT and implemented in AuthorIT's AutoBuilder component, make it possible to represent declarative as well as procedural knowledge, expert as well as neophyte knowledge along with all gradations in between. It also provides a way to represent model-based knowledge, which involves declarative knowledge about and procedural knowledge in a system together with inferential knowledge (currently limited to forward chaining) involving various components of the model (cf., Scandura, 2003).

Given an AST-based representation of content, TutorIT makes all instructional decisions based entirely on the AST structure representing the content, without any custom instructional logic (associated with content semantics). Moreover, AuthorIT's Configuration Tool can be configured so TutorIT delivers instruction in any number of ways, ranging from simulation to highly adaptive, including multiple variations on self-directed learning, practice, diagnosis and instruction.

Structural (Cognitive Task) Analysis (SA), as defined in Scandura (2003), and realized in AuthorIT (2005), provides an explicit basis for identifying ASTs in which each level represents equivalent knowledge at a different level of abstraction (cf. single level directed graphs/flow charts used earlier, Scandura et al, 1971, 1973, 1977). Different levels of abstraction directly reflect degrees of expertise, and TutorIT explicitly attends to these levels in directing instruction.

Expert knowledge of instructional design, for example, can be represented at the highest level of abstraction by naming an input-output structure, which represents the behavior to-be-constructed (a design for

instruction), and a process “design instruction”, which represents the associated procedural knowledge for creating such designs. Given a body of content, the expert simply “knows” how to design an instructional system for teaching that knowledge. Knowledge at this level of abstraction is largely structural/declarative in nature. Details of the process need not be specified. The expert performs tasks quickly, and TutorIT accordingly does not attend to details of the “automated” processes used (by experts). Neophyte knowledge involves more detailed specification of the instructional design process. SA provides a systematic level-by-level method for deriving such specifications in as much (or as little) detail as may be desired (e.g., see Scandura, 2003, 2005).

To date, AuthorIT has been used to build tutors in a range of tasks from arithmetic to higher order mathematical processes. While it can certainly be made more robust and further improved cosmetically, current support in AuthorIT is quite adequate for developing a wide range of highly adaptive, customizable and practical ITS systems.

More important here, the prototypes developed to date demonstrate that such development can be accomplished by moderately trained personnel at a fraction of traditional costs (e.g., half of the actual costs experienced by PLATO Learning, Foshay & Preese, 2005, pp. 251-2).

TABLE
Estimated AuthorIT vs. Current PLATO Time Allocations.

| | |
|---------------------------------------|-------------------|
| Front end analysis | – no change (20%) |
| Instructional Strategy Definition | – 1% vs. 30% |
| Creation of Media and software assets | – no change (20%) |
| Integration of components and testing | – 5% vs. 30%* |

*Continuing reductions are expected as the system matures.

Because AuthorIT includes an explicit method of development, each of these categories is defined below:

- Front End Analysis: identifying what needs to be learned, with whatever degree of precision might be desired — ranging from

informal to very rigorous. It also includes identifying (e.g., typing in) a question, instruction, positive and corrective feedback for each node in an AST representation (e.g., for approximately 20 nodes in a full characterization of column subtraction).

- Instructional Strategy Definition: Filling in a dialog box to define one or more delivery modes — including highly adaptive, pure instruction, pure diagnostic, simulation/performance aid or just practice. One can also pre-define such things as learner entry assumptions for each node (+, - ?), mastery criteria, when testing and instruction are to take place, and presentation, response and evaluation modes.
- Creation of Media and Software Assets: Simple things like text, simple graphics, OLE objects are attached directly to AST nodes and fully supported within AuthorIT. These may be supplemented by attaching sound files (for or in addition to the text), Flash files for animations, etc. The latter are automatically "played back" when appropriate by Microsoft Sound Recorder, Flash reader, etc. Each node is considered individually — no transitioning is necessary as that is handled automatically.
- Integration of Components and Testing: Assuming no remaining "bugs" or other one-time adjustments or enhancements needed in AuthorIT or TutorIT, no testing is necessary except to see the whole all at once for possible esthetic or other desirable adjustments.

Foshay and Preese may have used slightly different definitions. In this case, within as well as between category comparisons could play a role in calculating actual savings.

At the present time, AuthorIT does not distinguish well-defined and ill-defined problem solving. Both lend themselves to the same form of analysis (in AutoBuilder). Like traditional ITS systems based on production systems, however, solving novel or otherwise unanticipated problems is handled by simple chaining. Given the documented importance of so-called "situated" learning, it also should be emphasized that ALL ASTs have explicit domain and range structures. Defining input structures automatically designates (situated) domain of applicability.

Apart from AutoBuilder, the method of SA itself is directly applicable to higher order ASTs, and has been used manually to identify higher as well as lower order knowledge associated with problem solving in arbitrarily complex domains, including ill-defined domains (e.g., Scandura, 1973, 1977. Scandura & Scandura, 1980). The challenge is to extend AuthorIT

and TutorIT to accommodate ill-defined problem solving, by building on this theoretical foundation (see universal control below).

As described in Scandura (2005) planned but not yet realized goals include:

- allowing learners to create their own problems,
- extending support to include more powerful response and evaluation types.
- adding support for ill-defined problem solving – the latter requiring implementation and refinement of a patented universal control mechanism and
- extending support for collaborative learning.

Despite these capabilities and broad applicability, AuthorIT does not negate the usefulness of alternative approaches to authoring. Taxonomic approaches have traditionally been easier to put into practice, and simulations used to promote problem solving skills have already been used to good effect. What AuthorIT promises is to make many of these tasks easier and more cost effective – many now and more later as AuthorIT is subjected to the stress testing of everyday use. Another major benefit is the availability of multiple delivery/learning modes at no additional cost in time or money. The value of AuthorIT in practice, of course, can only be determined thorough actual use. This first and foremost is why both academics and innovative commercial operations are invited to put AuthorIT to the test.

REFERENCES

- Foshay, W. R. & Preese, F. Do we need Authoring systems? A commercial perspective. *Technology, Instruction, Cognition & Learning (TICL)*, 2005, 2, 3, 249-260.
- Scandura, J.M. AuthorIT: Breakthrough in authoring adaptive and configurable tutoring systems.. *Technology, Instruction, Cognition & Learning (TICL)*, 2005, 2, 3, 185-230.