

TICL 2: Knowledge Representation, Associated Theories and Implications for Instructional Systems Dialog on Deep Structures

AERA Tuesday, March 25, 2008

JOSEPH M. SCANDURA, KEN KOEDINGER, STELLAN OHLSSON,
ANTONIJA MITRTOVIC AND GILBERT PARQUETTE

INTRODUCTION

This discussion forum is designed to compare and contrast major alternative deep infrastructures on which progress in TICL depends. This issue has the goal of increasing understanding, specifically by exposing and clarifying basic similarities and differences between three invited articles in TICL Vol. 5, No. 2. Each focuses on a different basic approach to deep infrastructure in TICL. The Ohlsson & Mitrovic article focuses on production systems (PS) and constraint based modeling (CBM), and their use in building ITS. Paquette focuses on instructional systems derived from instructional design principles and relational networks. Scandura focuses on Abstract Syntax Trees (ASTs) and the central role they play in Structural Learning Theory (SLT) and in building adaptive and configurable tutoring systems derived thereon.

The ensuing discussion begins with a dialog involving the three sets of authors, and those they have directly challenged. In all cases the goal has been both clarification and comparison. Initially the dialog focuses on comments and responses about the three core articles, by authors themselves and/or those whose ideas they may have challenged. Once these authors have had their say, probing questions, insightful comments and reasoned criticisms based on or deriving from the articles and ensuing discussion will be welcomed from informed researchers who have read the articles and the above commentary. Questioners will include both other authors in the special triple issue, and invited questioners at a forthcoming 2008 TICL symposium in New York. All are leaders in the field.

Both the initial and this follow-on issue were planned and executed in the belief that true progress in TICL can only, or at least can best be achieved only by clarifying the many similarities and differences that both characterize and retard progress in our field. These similarities and differences have often gone undetected – sometimes for decades – often hidden in different terminology or overlapping ideas. Hopefully, the ensuing dialog will help clear the path to more rapid, cumulative advances in the future.

KEN KOEDINGER 1: RESPONSE TO OHLSSON & MITROVIC

The Ohlsson and Mitrovic article contains two fundamental inaccuracies regarding model-tracing tutors:

1. It states: “No version of the ACT theory ever contained a learning mechanism that can learn from feedback in response to an error, i.e., from the primary form of instruction delivered by the model-tracing tutors that have been published to date”.

However, “feedback in response to an error” is not the “primary form of instruction delivered by the model-tracing tutors”. Ahead of it are the next-step hint messages the model-tracing tutors provide on request from the student and, if the student is really stuck, the bottom-out hint messages where the tutor tells the student the next step. Students are aided to eventually generate correct solution steps and these are the basis for learning by example. The ACT theory has always contained a learning mechanism for learning by example as acknowledged by the authors.

2. The section on “Efficiency of implementing an expert model” fails to mention the important trade-off that a constraint-based expert model does not generate next-step hints whereas a model-tracing expert model does. If a tutor developer puts in the extra work to add the capability of generating next-step hints to a constraint-based expert model, they are likely to have done at least as much work as in developing a model-tracing tutor.

OHLSSON & MITROVIC 1: RESPONSE TO KOEDINGER

In response to our claim that there is no connection, and never has been any connection, between the learning hypotheses in the various versions of the ACT

theory, on the one hand, and the mode of operation of the various model-tracing tutors claimed to be “based on” that theory, Ken Koedinger says that the main mode of instruction in model-tracing tutors is to prompt the student to do the right thing, which allows him or her to learn from example via analogy or generalization.

If model-tracing tutor teach by prompting the student to do the right thing, and, when that fails, telling them the right thing to do, why would those tutors need to model trace? In that mode of tutoring, there is no need for any diagnosis of the nature of the student’s error when he or she does the wrong thing. So why implement a library of buggy rules? Published model-tracing tutors have been described as containing in excess of 400 rules, laboriously constructed through as much as 10 man-hours per rule. Why invest this effort, if the main mode of instruction is simply to say “do X!” when the student should have done X but did Y? On the theory that students learn by analogizing or generalizing from their own correct steps, model tracing is not needed, and the fact that model tracing tutors contain production rules is irrelevant, severing the last link between the ACT theory and the tutoring systems supposedly based on it.

Koedinger’s statement does not correspond to the published rationale for the model-tracing technique, but presumably it accurately describes where the CMU tutoring enterprise has ended up in practice. The question then arises whether this is a good theory of tutoring that provides useful guidelines for the design and implementation of ITSSs. The hypothesis that students only learn from their correct steps is flatly contradicted by the success of constraint-based tutors, because the latter instruct primarily by helping students correct their errors and students learn from interacting with those tutors. The sane position is of course that people learn in different ways, through different cognitive mechanisms, and that more powerful tutors will result from providing support for several different modes of learning.

With respect to efficiency of implementation, Koedinger brings up the fact that adding next-step hints to constraint-based tutors would increase the amount of work needed to implement such tutors. Yes, but so what? Constraint-based tutors do not need specific next-step hints; that is one of the strengths of the constraint-based technique. This is good, because next-step hints can be difficult to provide for many real-world domains that lack well-defined solutions. It is a weakness of the model-tracing approach that it relies heavily on such hints. CBM tutors can do without them, so Koedinger’s hypothetical comparison of the work loads that would result if one were to add next-step hints to a CBM tutor is irrelevant. In short, on neither point does our article contain “fundamental inaccuracies.”

SCANDURA 1: BROADENING THE DISCUSSION - RESPONSE TO KOEDINGER AND OHLSSON

Without prejudging Ken's (KK) response, I would like to broaden the conversation a bit. Specifically, I question whether it is desirable, let alone necessary to make assumptions about what is actually going on in learner brains.

Neural Processes. I agree with Stellan that constraint-based modeling (CBM) moves production system based tutoring in a positive direction. Rather than trying to adjust instruction based on assumed actions (productions) in the brains of individual learners, CBM starts with assumptions about declarative constraints used by learners to assess applicable situations and their satisfaction.

CBM eliminates the need for biologically based assumptions about what constitute atomistic actions (i.e., productions) by focusing on declarative knowledge. Nonetheless, production systems (PS) still represent a core assumption (O&M, p. 118). On page 126 O&M add that: a) production rules as a representation of procedural knowledge have an edge over all other forms of KR and b) CBM has similar advantages for declarative knowledge. This statement presumably derives from the belief that production systems and constraints are not only fully executable, but also directly reflect knowledge in learner minds.

As O&M note (pp. 122–3) worrying about buggy productions in developing ITS depends on time consuming and expensive empirical study – and serves no real purpose. As 30 years of work with ITS makes clear this is not an easy task. While O&M argue that identifying constraints takes less work, CBM also requires a good deal of guesswork. And, this is to be expected. Identifying the knowledge potentially available to learners requires making numerous fundamental assumptions about the atoms of knowledge available, whether procedural as in PS or constraints as in CBR.

While such assumptions seem appropriate where the goal is to model the brain, I seriously question whether this is the best, or even a good way to design “intelligent”, highly adaptive tutoring systems. Trying to pre-determine the ingredients that any given learner brings to any non-trivial part of mathematics, for example, or poetry for that matter, seems at best improbable. We will never be able to know with any certainty, much less predetermine, what elements of knowledge are available to any individual, let alone to any population of learners at any given point in time.

Although we can program computers, we cannot program much less debug humans. As educators we don't have license to look inside (cf. Scandura, 2001, p.312). Basing diagnosis and instruction on such assumptions is analogous to

treating the common cold by reference to manipulating chemical reactions in the body. Theoretically possible – perhaps – but hopelessly complex in practice.

My article demonstrates how subject matter experts can systematically identify the knowledge needed to master any given domain – with any desired degree of precision. The article also shows how this knowledge can be used to determine what any given learner does and does not know about what the expert(s) know, and how to help learners achieve similar mastery. These and related issues are elaborated below.

Learning Mechanisms. As O&M emphasize, progressive versions of ACT have postulated various numbers and kinds of learning mechanisms. Historical roots may be traced to Newell & Simon's (1962) means-ends analysis. CBM shares with ACT (and other PS-based tutoring) the assumption that learning mechanisms play a fundamental role in designing tutoring systems. O&M are less definitive about learning mechanisms, however, stating that they don't presume to know what learning mechanisms are in the learner's mind (p.108). (Why then is it important to worry about what productions, or constraints might be in the learner's mind?)

Furthermore, O&M make no claim to know what learning mechanisms will be most helpful. Their arguments are based on plausibility, generality, sufficiency and adequacy with respect to available data. O&M (pp. 108–9) suggest that there may be any number of learning mechanisms, depending on the type of input (i.e., the domain in question).

Accordingly, O&M propose that “the natural mode for instruction in an ITS is to provide help when the student is making errors” (p. 110). Ohlsson 1 further argues that learning mechanisms postulated in ACT have little impact on either instruction or corrective feedback in Carnegie's recent model tracing tutors (cf., Ritter, 2005). In both cases, the main pedagogical tactic is telling learners the “right thing to do”.

O&M do not claim that CBM provides a complete foundation for all tutoring. They specifically suggest that future research is needed to determine the extent to which any given learning system supports the full range of ways (i.e., mechanisms) by which people learn (O&M, p.128). (NOTE: My research strongly suggests that these mechanisms may be characterized in terms of the higher order rules associated with any given domain.)

My Concerns. While I agree CBM is likely to produce comparable results to PS at less cost, I question whether either goes far enough. Rather than worrying about what productions or constraints may be in the minds of learners, why not concentrate first and foremost on what must be learned – and, secondarily on what prerequisites are necessary for a learner to benefit from associated tutoring?

Similarly, rather than pre-judging which basic learning mechanisms will be most important, why not consider the alternative of systematically identifying relevant mechanisms (higher order SLT rules) from the domain itself. O&M come close in saying that learning mechanisms depend on the type of input (i.e., constraints), as does their statement “efficient retrieval and application of knowledge requires a certain amount of organization” (p. 108). Exactly what are these learning mechanisms? Despite 30 plus years of (traditional) empirical research, by any account the jury is still out.

Resolution. I have proposed answers to each of these questions: 1) Rather than asking subject matter experts (SMEs) to identify the basic ingredients that might go into a knowledge base, why not ask SMEs to more simply identify what they believe must be known for success (e.g., pp. 173–5; 181–94)? As experts, SMEs know how to solve most if not all problems in any given domain. What is needed most is a way to help them represent what must be learned to solve problems in a way that can be implemented on a computer. The method of Structural (domain) Analysis was constructed with SMEs explicitly in mind (pp. 175, 194–216). (NOTE: SMEs as in Structural (domain) Analysis tend to work from the top-down, asking what (e.g., components) are needed, or what they would like to have or know to solve given problems. Knowledge engineers are asked to identify the equivalent of programming languages (e.g., made up of productions) sufficient for solving any problem. Production based systems require assembling those components to achieve perceived ends (i.e., work from the bottom up). In effect, SMEs are more like designers; Knowledge Engineers, more like programmers.)

Where is the learner (model) in this scenario? To teach something efficiently it is not sufficient to know only what to teach. We also need to know something about what the learner does and does not already know. In PS modeling and CBM this is accomplished by making assumptions about ingredients (productions or constraints) in the learner’s brain – and then making further assumptions about how those ingredients are used, combined and/or modified in solving given problems.

Why do this when the only thing we can reliably observe is the learner’s behavior? In SLT, what any given learner knows is operationally defined relative to what one is trying to teach (e.g., pp. 177–8, 231–41, 192). Systematic (automated) selection of problems and sub-problems (corresponding to constraints) makes it possible to very efficiently identify what any individual knows at each point in time (relative to what SME’s believe should be taught). I propose that one cannot tell the difference by observing a learner’s behavior between what is in the learner’s brain and measuring the learner’s behavior potential relative to some ideal. Why is it better to make questionable assumptions about what is actually

going on in a learner's brain? To teach something, what do we need to know beyond what the learner can and cannot do that is relevant?

Nonetheless, in its simplest form, this solution is incomplete. It is not possible to identify solutions (SLT rules for solving) all, or even most problems in non-trivial domains. The numbers and varieties of problems may be indefinitely large (cf. Scandura, 1971, 2007). Unanticipated (i.e., novel) problems in such domains may be solved in PS modeling and CBM via assumed learning mechanisms by combining and/or modifying existing productions or constraints. (NOTE: An alternative, top-down (ICL) approach is to assume one or more learning mechanisms in conjunction with some kind of problem taxonomy. In case based reasoning (CBR) (e.g., see Jonassen, this issue), for example, the goal is to identify and devise solutions illustrating various types of problems – in the hope that learners will use built-in CBR learning mechanisms to generalize to other problems. Similar arguments can be made in the case of means-ends analysis, chaining, abstraction, etc. – or, chunking in the case of automation – p. 190.)

Exactly what are these “learning mechanisms”? In general terms learning mechanisms correspond to the kinds of (higher order) heuristics introduced years ago by Polya (1960). He identified a variety of heuristics, and illustrated their role in solving complex mathematical problems. He also demonstrated, however, that there are any number of heuristics. While various heuristics may share common features (e.g., generalization, chaining, etc.), they typically also have characteristics unique to the particular domains from which they were derived. In effect, there can be as many learning mechanisms as there are domains – more since any one domain may involve any number of learning mechanisms (which I have called higher order SLT rules, pp. 188–90; 204–12).

Such mechanisms come with no guarantees as to whether they will work. We have a choice here. Either accept that the best we can do is identify general classes of learning mechanisms – leaving considerable discretion as to how they are implemented (i.e., made executable). Or, turn things around, and simply view learning mechanisms as (learnable) knowledge – albeit higher order knowledge that depends in varying degrees on the domain in question. In the latter case, we still have the problem of explaining how does learning take place?

Proposed solutions to the above were detailed in my article:

1. Systematically identify not only solutions (I called them SLT rules) for solving prototypic problems in given domains, but also the learning mechanisms (higher order SLT rules) derived from those SLT solution rules (pp. 175, 194–216).
2. Assume that all learning and behavior is governed by a single fundamental mechanism, which applies universally, not only to problem solutions (SLT

- rules) but to traditional learning mechanisms and their variations (what I have called higher order SLT rules, pp. 175–7; 216–31).
3. Operationally define what any individual knows relative to lower and higher order to-be-learned SLT rules (pp. 177–8; 191–3; 231–41).

In this context, I raise the following key points for discussion and debate as to their scientific validity and use in practice.

Unlike procedural PS and declarative CBM, SLT rules based on Abstract Syntax Tree (AST) representations inherently include both. It is not a question of whether procedural or declarative representations are better. Like every computer program ALL knowledge elements in SLT (SLT rules) necessarily involve both declarative and procedural knowledge. Every executable requires both a data structure and a procedure, which operates thereon. The relative mix in each case is just a matter of degree. (In retrospect, it is a mystery as to why it has taken the tutoring community so long to capitalize on this basic fact.) Why restrict knowledge to mechanisms operating on productions (as in production systems), or on structural constraints (as in CBM).

SLT rules based on arbitrarily refinable ASTs (pp.184; 195–9) have the additional advantage of representing various levels of expertise in single representations, something that is not easily accomplished with either production systems or constraints. (NOTE: Arbitrarily refinable ASTs take O&M’s statement about the need for organization [p. 108] to its natural conclusion.) This fact makes it quite simple, in fact trivial, to represent essentially any knowledge, no matter how complex with some degree of specificity. Many would throw up their hands in despair if asked to construct a program that would teach students how “Given an idea, to write a beautiful poem (about it)”.

Complexity and ambiguity aside, one would use exactly the same method of structural analysis (SA) in this case as in computational arithmetic. Unlike arithmetic, of course, poetry experts may not agree on exactly how best to write poems when it gets down to specifics. Indeed, like any design problem, there is undoubtedly more than one way to proceed. Nonetheless, SA makes it possible to systematically identify the associated higher order knowledge, whether this involves generating new SLT rules as needed, or selecting from alternatives (pp. 194–216).

What biologically inspired theorists call learning mechanisms (i.e., higher order SLT rules) also are constructed systematically (via SA). How to systematically and reliably identify higher order SLT rules posed a serious dilemma for many years. Although most/many agree on the importance of certain basic types (e.g., chaining, generalization, abstraction, selection, automation, etc., p. 190), in fact there are any number of possible variations of each depending on the domain in question.

If higher order knowledge can vary so widely, what is it that makes behavior and learning possible? If no “hard wired” learning mechanism is sufficient, how then is learning to take place? What determines when higher as well as lower order SLT rules are used? And, how are they learned?

While a universal “goal switching” mechanism was originally proposed in Scandura (1971), it was not possible for many years to formulate this control mechanism as an executable that is completely independent of the (multiplicity of) higher order SLT rules associated with various domains. We have recently succeeded (see pp. 175–7, 216–31) in reformulating this Universal Control Mechanism (UCM) in a way that meets this requirement (cf. Wulfeck & Scandura, 1977 where UCM was inextricably intertwined with the higher order rules involved).

Indeed, universality is crucial if one is to accommodate arbitrary, domain influenced higher order SLT rules (aka learning mechanisms). Clearly, while current thinking presupposes multiple mechanisms, PS and CBM theorists presumably would be very unhappy if forced to have a different hard-wired mechanism for every domain that might be of interest.

What is needed is a single UCM – one that works not only with arbitrary SLT rules but also with the equivalent of essentially any traditional learning mechanism (or variation thereof). To the extent that UCM can, in fact, be implemented, any higher order SLT rule (the equivalent of a learning mechanism)” might be plugged in – without change to the basic architecture. Despite strong empirical support for informal versions of UCM, going back to the early 1970s (pp. 217–8), its importance has been overlooked for many years both because it has never been fully automated – and because it demands viewing familiar phenomena from an entirely different perspective.

My final challenge pertains to perceived differences between what is actually in a learner’s brain and observable behavior. Let us assume that an equally good job has been done in: a) identifying productions or constraints in the brain and b) what must be learned for success. The question is how would one distinguish between the two? We present a problem, and the learner responds. Assume the learner is correct. In the first case, we explain the behavior in terms of interactions among the components. In the second, we explain the behavior by identifying parts of the to-be-learned knowledge required for success.

Now, assume the learner is incorrect. In the first case, we explain the behavior either in terms of buggy productions or constraint defects. In an ITS one first identifies what production(s) or constraint(s) are missing and/or defective. Then, the ITS provides appropriate corrective feedback or instruction.

In SLT, we explain the behavior by observing that the learner has not yet mastered something that domain experts believe is essential. It is a trivial matter to

identify what is missing – those parts of to-be-acquired SLT rules that the learner does not already know. The fundamental question is: **Does it make any difference what is (or is not) actually in the learner’s mind, if what we want is to help learners master what is to be learned?** I propose that the desired (to-be-taught) behavior is the same in both cases – and, so is the necessary instruction.

There is still more. Tutoring requires deciding what to teach next. This is costly in ITSSs based on PS or CBM – precisely because neither PS nor CBM includes a general, semantics-free method for organizing productions or constraints. Accordingly, I am unaware of any general way to make instructional decisions in either PS or CBM without explicit reference to semantic characteristics of the to-be-learned content. Counterexamples are explicitly welcomed of course.

AST-based knowledge representation, on the other hand, makes diagnosis and instruction possible by exposing the structure of what is to be learned – independently of its semantics (pp. 178–80; 241–51). In SLT, learning mechanisms are not “hard wired”, but rather correspond to learnable higher order knowledge. In effect, diagnosis and instruction is accomplished in the same manner, whether the SLT rules involved are of higher order or lower order.

OHLSSON 2: RESPONSE TO BROADENING THE DISCUSSION IN SCANDURA 1

Comment on “Scandura 1 - Broadening the discussion”: I am sympathetic to the general trend of Scandura’s response to the O&M paper (Scandura -1), which I take to be that the power of psychological learning theory, especially in computational form, to inform instructional design has so far been limited. Looking back over arithmetic manipulatives, falsification tactics for triggering conceptual change, “whole word” reading, and so on, I tend to agree. I also agree that there is a limit as to how detailed any model of an individual student can be, and that no instructor, human or machine, can ever know the student’s knowledge base in all the details that might be required for deciding what is exactly the right instruction to deliver next. But your alternative seems to be to focus on the subject matter: “Rather than worrying about what productions or constraints may be in the minds of learners, why not concentrate first and foremost on what must be learned?” But this is the oldest instructional design fallacy of all, to believe that the structure of the subject matter, by itself, implies a particular way to learn it or teach it. This was precisely the idea behind the so-called “new math”, the attempt to base math understanding on set theory; that looked great from a subject matter point of view and was pushed by subject matter experts, ie mathema-

ticians; it was a disaster. The history of education is littered with textbooks, labs and exercises invented by brilliant subject matter experts but unsuccessful in teaching students anything.

I take the purpose of cognitive work on learning and instruction to be going beyond what can be done by merely looking at the subject matter itself, and also take into account how it is to be learned. It would be exceedingly strange if knowledge of the mind's learning mechanisms held NO IMPLICATIONS for how a particular subject should be taught. An analogue would be that knowledge of the disease mechanism behind a particular symptom would turn out to hold no implications for how the disease is to be, or can be, treated.

To resolve this paradox, I think we have to make a distinction between specific points like "learning theory X generates implications so-and-so, and they turned out not to work" and the universal point "all learning theories generate implications that do not work." The universal point ignores the fact that some theories are good and some are bad; so far, the impact of psychological learning theories have largely been negative or neutral, which I take to mean that those theories are wrong. We should not expect a false theory to generate effective implications for practice. But this fact has no implications for whether the right theory, once we find it, will generate effective implications or not. The remedy is to formulate smarter, better empirically grounded learning theories and test their instructional implications. After all, treatments based on the humour-balance or miasma theories of disease were not effective either; medicine didn't become effective until after the germ theory won out.

So where does CBM stand in this view? The key component of CBM as far as ITS is concerned is the constraint notation/representation and the simple pattern matching technique for applying constraints to student solutions and inputs. This technique for how to give feedback about errors and mistakes stands on its own. That is, there is a learning theory that shows how the mind can learn from constraint violations. For anybody who believes in that theory, that's a strength. For someone who is skeptical of that theory, the CBM technique can still be the tool of choice. There is nothing in the technique itself that presupposes that the learner's mind changes in precisely the fashion claimed by the constraint-based specialization algorithm, which is the core of the associated learning theory. All you need to believe is that people can learn from their errors, and that to do so they need to be aware of them. The CBM technique allows you to specify which domain behaviors are to count as pedagogically significant errors and specifies what to say when your system sees those errors in student behavior.

But what about the claim that in applying CBM, we have to know the details of the learner's mind, and we'd be better off focusing on the subject matter? This

is no objection to CBM. The constraint base for a domain IS a representation of the subject matter. The constraints are meant to encode correctness for the domain. It is most natural to encode it this way if you also believe that people encode (some of their) declarative knowledge that way, but even without that belief, it makes perfect sense to encode domain knowledge as a set of constraints. The constraint base corresponds to the “expert module” or “ideal student” in other types of tutoring technologies. That is, it is something a subject matter expert, not a psychologist, is most competent to implement, same as an expert module in a rule-based system. In fact, that is the one of the main points of CBM: There is no need for empirical studies of students. The constraint base encodes the analysis of the subject matter, and then directly provides the functionality of being able to diagnose student errors without also modeling the student in a separate model.

From this point of view, CBM is a tutoring technology that conforms very nicely to the subject matter based approach Scandura outlines in his comment on the O&M paper.

SCANDURA 2: RESPONSE TO OHLSSON 2, REACTION TO BROADENING THE DISCUSSION

It’s reassuring to see that Ohlsson agrees that there are limits as to how detailed any student model can be. I also agree that what is most essential from an instructional point of view is to identify “exactly the right instruction” at each point in time.

I believe, however, that Ohlsson misinterprets some critical ideas in SLT and/ or their relationships to CBM. Listed below are the most important issues.

1. The focus on “what must be learned” is definitely not the same as subject matter as traditionally presented in textbooks. What must be learned refers specifically to executable competence that must be acquired to perform successfully on problems in some given subject matter domain. For example, it is not a question of simply teaching formal set theory to children as a prelude to arithmetic (and simple algebra) as in the “new math” of the 1960s. Rather, it involves first identifying a domain of problems (which may include anything from simple tasks to complex novel problem-solving). As emphasized in my very first article outlining SLT (Scandura, 1971), nontrivial domains will invariably include indefinitely large numbers of problems – including novel problems that may initially have been unanticipated.

Clearly, identifying domains that have relevance in school mathematics (or any other subject matter) demands both knowledge of the subject matter and a strong sense of important educational objectives – what I have referred to as “subject matter expertise”.

(NOTE: Ironically, I received my Ph.D. in mathematics education during the heyday of the “new mathematics” in the 1960s. My early research during this period helped initiate a trend toward more basic research on mathematics learning [Scandura, 1966], from which SLT later evolved.)

2. A second equally important difference is in how “what must be learned” is viewed in SLT, as a set of lower and higher order SLT rules, and in CBM, as a set of constraints. In the former case, Structural (domain) Analysis provides a general and highly systematic method for identifying SLT rules, which collectively are sufficient for solving arbitrary problems in the given domain. Identifying appropriate constraints is one of the most essential, yet subjective requirements in CBM.
3. I also very much agree with Ohlsson that cognition plays an essential role. Our differences here derive from Ohlsson’s apparent belief that learning mechanisms are somehow built in (and apparently immutable characteristics of the brain). In SLT what others call learning mechanisms are represented as higher order SLT rules, which are associated with given domains (whether well-defined or large and ill-defined). Higher order SLT rules can be systematically identified and learned independently. One can identify higher order rules corresponding to essentially any learning mechanism, ranging from Newell & Simon’s (1972) historical means ends analysis to chaining, analogy (case based reasoning), generalization, logical inference, selection and/or automatization (chunking). The basic confusion here derives from the belief in CBM (and production system theories generally) that each of these “kinds” of mechanisms is universally applicable and independent of the subject matter (or, more generally, content domain). Structural Analysis demonstrates that there are variations on each. Different variants may have different domains of applicability.

Viewed as a learning mechanism, for example, simple chaining is universally applicable. In fact, however, direct experimental evidence demonstrates that many grade school children are unable to solve “A-?C” (e.g., 3 yards = ? inches) problems after they have learned “A->B” and “B->C” rules (e.g., 1 yard = 3 feet and 1 foot = 12 inches). Experiments show that some learners chain rules (in some domains) but not in others (e.g., Scandura, 1967). The same is true of logical inference. Thus, Modus Ponens (the simple syllogism “If A, then B” and “A” implies “B”) is universally applicable from a mathematical point of view. On the other hand, recognition of

- its applicability in concrete situations is definitely context dependent (e.g., Lowerre & Scandura, 1974).
4. Having said this, Ohlsson is certainly correct in that cognition plays a role in learning. Rather than introducing an indeterminately large number of learning mechanisms, however, cognition in SLT is based on a single universally applicable control mechanism (UCM). UCM controls the use of all SLT rules, whether of higher or lower order.
 5. I also agree that constraints play an important role in characterizing the “ideal student”. In SLT, these constraints correspond to the declarative/structural aspects of SLT rules (i.e., the expert model characterizing what must be learned). Although constraints tell part of the story, however, they leave out an equally (if not more) important part of what characterizes the “ideal student” – namely procedural knowledge. I also agree that introducing constraints eliminates the need for empirical studies to identify productions (including or excluding error productions) that students might use. What constraints do not do is include the procedural knowledge that an “ideal student” must have.

In effect, use of constraints eliminates the need to know how the “ideal student” got there. What it does not do is tell the tutor what the “ideal student” needs to know (to progress to the next stage of learning).

Summary. Having pointed out the above differences, what impresses me most is what appears to be a mutual convergence. Working from the bottom up, Ohlsson and Mitrovic have developed CBM in reaction to limitations in traditional production system based theory. Conversely, SLT has evolved by working from instructional principles, largely from the top down with important parallel developments in software engineering (which provided the necessary technical foundation).

REFERENCES

- Lowerre, G. & Scandura, J.M. Development and evaluation of individualized materials for critical thinking based on logical inference. *Reading Research Quarterly*, 1973, *IX*, 185–205.
- Newell, A. & Simon, H.A. *Human Problem Solving*. Englewood Cliffs: Prentice Hall, 1972.
- Scandura, J.M. (Ed.). *Research in mathematics education*. Washington, DC: NCTM, 1967.
- Scandura, J.M. Deterministic theorizing in structural learning: Three levels of empiricism. *Journal of Structural Learning*, 1971, *3*, 21–53.
- Scandura, J.M. The role of higher-order rules in problem solving. *Journal of Experimental Psychology*, 1974, *120*, 984–991.

PAQUETTE 1: STRUCTURING ROLES OF COMPETENCY: RESPONSE TO OHLSSON 2 AND SCANDURA 2

I find it hard to reconcile my approach with both Ohlsson-Mitrovic and Scandura papers. They focus on automatic tutoring while I have strong suspicion against it, except in very specific contexts. I have yet to see an efficient ITS outside unconstrained domains like all of the examples given in the two papers. A second problem is that generally, ITSs guide the student too closely depriving them sufficient “liberty to learn”. Finally, it is hard for a software agent to make sense of what a learner is doing, as it is has limited perception. Of course, a software agent, properly prepared, can be endowed with structured knowledge and use it more systematically and logically than a human generally does.

This doesn’t mean that I refuse to build knowledge-based systems that provide some automatic help to student. But I situate this within other sources like a human tutor or other learners. In fact, in the past, I have built such systems to help designers or learners achieve their task, but my main goal is to build learning environments that are multi-agent systems, where human and computerized agents interact.

Target competency and learning processes

In that context, I will now focus on the learner assistance agents that should be provided in a knowledge-based environment. In my paper, I did not address this question directly, being more focused on the use of knowledge representation for the design of well-structured learning environments. But I will start from there to address the tutoring question.

First of all, in a learning unit, there should be a clear and operational goal that I call a *target competency*. Our instructional method (MISA) is based on that simple premise: if you don’t have such a goal, or if you have one and cannot do anything with it for the design and delivery of knowledge, then you should not be teaching at all; your student will do some free-for-all activities where they might learn something as in everyday life, without being disturbed by some invader who contradict their natural ability to learn for some obscure reason. Of course, we can do better than that.

Target competencies are higher-order knowledge, in the sense that I believe Scandura defines it: “executable competence that must be acquired to perform successfully on problems in some given subject matter”. But they are not yet learning processes and strategies. I define a competency as a generic (cognitive, affective or psycho-motor) skill applied to domain (subject-matter) knowledge.

A *generic skill* is higher-order or meta-knowledge compared to domain knowledge. It is knowledge about knowledge, that can operate on any specific domain and is thus transferable to other domains. For example, to “know how to proceed in a information search on the Web” is not a clear target competency. No generic skill is involved. Is the goal to *simulate* or apply a given process, or to *improve* one, or to *build* (or design) one from scratch? *Simulate*, *improve* or *build* all denote generic skills that can be applied to other domains than information search on the Internet.¹

I also use the term *meta-process* to describe generic skills. It is interesting to see generic skills as processes and describe them by a process model (built using our MOT software). For example the *simulate* meta-process has inputs (a domain specific process to simulate) and an output (the trace of the simulation). It can be decomposed into sub-processes (provide input to the domain process, find a first task, execute it, find next ones, use simulation principles to provide new inputs, rerun, etc.). This is basically a generate-and-test process. Of outmost importance are the simulation principles that guide the execution of the meta-process. These provide a basis for learner guidance (or tutoring).

Cognitive Fidelity

I have read with great interest the debate between Ohlsson and Scandura on Cognitive fidelity, the first taking the cognitive psychologist view and the second the technologist view.

My position on this is two-fold. On one hand, I cannot agree with the fact that cognitive fidelity is irrelevant to build sound, useful and efficient learning environments and tutoring agents. Scandura states: “Basing diagnosis and instruction on such assumptions in analogous to treating the common cold by reference to manipulating chemical reactions in the body”. One can contradict that statement by considering the fact that no cure or relief is possible without knowledge about human physiology unless we rely on luck. The same is true about instructional technology compared to cognitive psychology and neurology, even though this is an even more considerable task. For example, the cognitivist and constructivist assumptions on human learning have considerable impact on the way we can built learning environments and tutors.

¹ I have built a taxonomy of such generic skills based on previous work by Bloom, Pitrat and KADS. This is explained in detailed in my books: Paquette, Gilbert, *Instructional Engineering for Network-Based Learning*, Pfeiffer/Wiley Pub. Co, 262 pages, December 2003; and more extensively in French: Paquette, Gilbert - *Modélisation des connaissances et des compétences, pour concevoir et apprendre*, 357 pp, Presses de l'Université du Québec, mai 2002.

On the other hand, the technologist stance by Scandura (“why not concentrate first and foremost on what must be learned”) is not, as Ohlsson states “the oldest instructional design fallacy of all that the structure of the subject matter, by itself, implies a particular way to learn or teach it”. It can be asserted that the words “first and foremost” are meant not to discard psychological assumptions to guide the design. Having been deeply involved in the new math movement in the 70’s, I must disagree with Ohlsson that it was a disaster and that it was dominated only by subject-matter consideration. The goal was also to provide knowledge on math deep structures. It included a proposal for problem solving approaches as a way to learn and teach math.

But if “what must be learned” is limited to “subject matter”, with the higher order rules being derived from it, then I must agree with Ohlsson that it is reductive. In my proposal, I make implicitly the hypothesis that learners have specific domain knowledge AND meta-processes (higher order rules) in their head. Both need to be improved. The way competencies are defined is that both level of knowledge (generic skills and domain knowledge) are being learned at the same time as learners solve problems or are instructed. This claim is well supported by the literature, both in education and cognitive science. But the meta-knowledge (learning mechanisms, higher-order rules) is not derived from the subject-matter. Both are orthogonal sets of knowledge that interact to define a target competency. They are orthogonal because generic skills are developed in a person’s history through numerous interactions with a variety of knowledge domains.

The equivalent of a UCM (Universal Control Mechanism) in my view is *the execution of the meta-process (generic skill) applied to domain specific knowledge*, which is exactly what a learner attempts when he/she tries to attain a target competency while interacting with a learning environment. I don’t know if such a UCM exist in the human brain, or if there are many, but we certainly need that one from an engineering point of view.

Learner Guidance

Let me now address the question of tutoring or *learner guidance*. Suppose I want to design a guidance agent to help learners improve their information search on the Internet. I could design it with two successive learning units: the first one aiming at simulating a pre-defined search process, and another where the learner would improve the now mastered search process, to improve it for more efficient search on the Internet.

One interesting way to do it is to structure the first unit around a scenario build directly from the simulation meta-process.

Correspondence between activities in learning scenario and tasks in generic process.

Activity in the scenario	Correspondence in the generic process
Activity 1: Choose a subject for the search	Inputs to the overall process, defining the case to be simulated
Activity 2: Open a browser and search engine	First two applicable procedure to execute
Activity 3: Build a first search request	Next applicable procedure to execute
Activity 4: Execute a search request	Execute the chosen procedure
Activity 5: Refine the first request if necessary	If unsuccessful redo previous activities
Activity 6: Transfer interesting information	If successful proceed to next procedure to execute
Activity 7: Report on your search process	Assemble the trace

Of course, the scenario in this table is not yet complete and its graph would present the relations between activities, some branching conditions and possible cycles with termination conditions. Also, a way to provide wire-in guidance would be to add resources that help, such as information on the structure of a request or on a final report form. Also, we might specify some collaboration rules between learners to achieve the activities. But the important thing here is that the generic process in the target competency becomes the backbone of the learner's assignments. In that way, we make sure that he exercises the right generic skill, in this case "simulating a process", while working on the specific knowledge domain (information search on the Internet), thus building specific domain knowledge and meta-knowledge at the same time.

Learner guidance is then a set of advices or actions in the environment associated to each activity in the scenario (tasks in the meta-process), and also to the global process (to provide help in its execution).

Advices can be implemented as productions, constraints, SLT rules or other methods. The important thing is that they implement control principles that guide the execution of the meta-process (generic skill). Other advices can be specific to the subject-matter domain, for example, how to use a query interface and a search engine.

Guidance can be given based on the meta-knowledge or on the domain knowledge. A carefully designed model (MOT graph) of the generic skill can uncover not only the meta-process part, but also the meta-principles (heuristics or control principles) that are associated to tasks in the meta-process.

I give here two simple examples of meta-principles for the generic skill “to simulate a given process”:

1. *if all the task in the simulated process have been executed, then assemble the simulation trace, else here are uncompleted tasks you should achieve.*
2. *if the simulated process has a branching condition and one branch has been successfully simulated, then select an input that will have another branch simulated.*

Such principles could be implemented as advices to learners using an ontology for scenario execution and another ontology to describe domain specific knowledge associated to activities and resources in the scenario.

CONCLUSION

Both Ohlsson and Scandura approaches are interesting. Ohlsson offers a simple, modular and efficient way to encode tutoring advice or actions at the local level. But in my view, constraints should be structured by the target competency. I mean by that that they would grouped in chunks and associated to activities at different levels in a scenario corresponding to sub-tasks of the meta-process (cognitive skill) in a target competency.

Scandura proposes, as I do, a two-level view. Domain specific knowledge is coded in the relation of SLT rules with AST (Abstract Syntax Trees). Meta-knowledge or learning mechanism is coded into higher level SLT rules. But I don't believe one can be derived from the other. They are orthogonal knowledge stores with multiple possible associations between them defined by target competencies serving as goals for a learning environment.

I don't have space here to discuss if constraints, SLT rules or ontologies are better to implement these constructs. There are certainly pros and cons as Ohlsson-Mitrovic discuss in their article. For my part, I have chosen to use the OWL-DL ontology language because it is an expressive way to represent knowledge, using a large subset of predicate calculus that is complete and decidable. My choice is also for engineering reasons. The work on the Semantic Web has entailed a growing stack of open source software development projects that can facilitate the construction of advisors or tutors for Web-based environments. I suspect that both Ohlsson's and Scandura's proposals could also be implemented this way.

SCANDURA 3: RESPONSES TO PAQUETTE ET AL

Paquette adds still another dimension to the discussion. Because Paquette tends to lump them together, let me first clarify essential differences between (all) ITS systems and SLT based tutoring systems (e.g., TutorIT, Scandura, 2005). Ohlsson and Mitrovick (OM) have modified the traditional production system approach to building ITS. Biologically inspired, both productions and constraints are viewed as analogous to neuronal complexes. Rather than making assumptions about productions in learner minds (whether correct or erroneous), however, OM argue that it is sufficient to identify constraints that affect behavior. (I am ignoring here finer grain distinctions such as the need/desirability of various kinds of constraints, such as operator, buggy and path constraints.) A related issue on which there seems to be some debate between the two ITS variants is whether one needs an ideal learner model – and if so exactly what it is. I return to this in a moment.

Regarding Ken Koedinger's critique, OM argue that error rules (in Model Tracing) add little, or nothing additional when it comes to instruction. As OM put it, if an ITS tells the learner the correct thing to do irrespective of whether or not the learner responds correctly, then why is it important to identify error rules? Time pressures prevented KK from responding given our short time frame. (His immediate response to OM was "makes one think!")

Both Model Tracing and CBM have a common major limitation. Neither provides anything close to a systematic, analytical way to identify the productions or constraints needed to teach any particular content. The lack of such a method makes the two approaches difficult to fairly evaluate (cf. Weitz et al, 2007, Mitrovic, 2007). In fact, I suspect that one can represent any production as a constraint (of some sort) and vice versa.

In contrast, Structural Analysis (SA) has been a mainstay in SLT from its inception's (Scandura, 1971, 1974, 1977). As detailed in my recent monograph (2007), the results of SA represent what must be learned – that is, an ideal learner model – what the subject matter expert believes the ideal learner should know. As also detailed therein, the results of SA provide an explicit and highly efficient basis for diagnosis and instruction.

As currently defined, productions are special cases of SLT rules (the latter representing conditions as structure ASTs). Constraints correspond to elements in structure ASTs in SLT rules, without corresponding procedures needed to generate behavior. Integrating these unstructured productions and constraints in the form of SLT rules has the effect of representing not just the cognitive elements themselves but also the way they must be structured (in the learner's mind) to generate desired behavior.

Equally important is how that knowledge is represented. Representing knowledge in terms of arbitrarily rigorous AST-based SLT rules has a major advantage. SLT rules make it possible to define essentially any pedagogy – based entirely on the structure of the knowledge representation, without concern for content semantics. This is simply not possible in Model Tracing or CBM. One cannot make decisions based on lists – certainly not without specific attention to the semantics of individual productions or constraints.

As MO emphasize identifying productions is very labor intensive and costly. They choose therefore to abstract away from internal productions and to concentrate on what is essentially declarative knowledge. SLT inherently deals with both. All individual knowledge, whether structural or procedural, is measured relative to what is to be learned. I believe that in principle (as well as practice) one can uncover everything that might be observed via the assessment procedures detailed in my monograph – with equal if not greater rigor and a lot less work.

Effectively, one has a choice:

1. Make sure that what is assumed to be in the brain is actually there (e.g., by elaborate training and/or experimental test) or
2. Infer what is in the brain from observable behavior measured relative to sets of idealized SLT rules representing what is to be learned.

To fully accommodate instructional needs, both Model Tracing and CBM approaches to tutoring will almost necessarily have to give more attention to what must be learned – identifying ideal learners. In ITS, the ideal learner currently consists of unstructured sets of productions or constraints. Model Tracing involves finding out how learners use these elements when solving problems. The fact that these elements can be combined in one way or another has led to what has sometimes appeared to be a growing array of learning mechanisms. This complication is avoided in SLT entirely because such mechanisms are represented as modular, systematically identifiable higher order SLT rules – all handled by a single UCM.

Let's now turn to Paquette's other remarks. Paquette states that he has never seen an efficient ITS, adding that "it is hard for a software agent to make sense of what a learner is doing". Nonetheless, he suggests that a software agent properly endowed with structured knowledge (as in SLT) might use that knowledge more systematically and logically than a human teacher. I clearly agree with him on this point.

Indeed, I look forward to demonstrating our Technologies. Defining diagnosis and instruction based on content structure, independently of content semantics, rep-

resents a significant step forward (with relevant patents received or pending). Among other things, our authoring and delivery systems, AuthorIT and TutorIT, are based directly on SLT principles, which dramatically reduce the cost of developing highly adaptive (and highly configurable) instruction.

Another unique feature of TutorIT is the ease with which it can be (re)configured, thereby alleviating Paquette's concern that learners are not given adequate opportunity to learn on their own. To the contrary, the Learner can simply choose to ask relevant questions or to receive whatever instruction he or she wishes at any time in an Open Learner Model.

SLT is completely general and supports any kind of instruction of which I am aware, whether it involves higher knowledge or collaborative learning (Scandura, 2005, 2007). On the other hand, as noted earlier, available technology at the present time supports only part of what is theoretically possible. The path forward is clear and detailed in my monograph (also see Scandura, 2005). As an Emeritus Professor, no longer with access to talented graduate students, I look forward to sharing relevant technologies and/or working with those who may be interested either in research or in helping to create highly adaptive and effective tutorials.

Having said this, Paquette's work builds on years of research and thinking in instructional design. He has introduced a variety of formalisms, all with the aim of representing content in a precise and executable manner. I am not surprised that Paquette agrees "What must be learned" is an essential starting point. Indeed, some form of task analysis has played a central role in Instructional Design for many years, starting with Gagne during the 1960s, and before him the USAF's Robert Miller in the late 1950s. Paquette takes this work to its natural limits – going well beyond the original focus on behavior, providing explicit formalisms for representing associated knowledge.

No ID approach, however, including Paquette's, adequately addresses the critical issue of adaptive instruction. ID focuses on presenting effective and efficient instruction. While testing also is a concern, testing tends to be either of the summative or pre-test variety. ID generally does not do a good job at integrating diagnosis and instruction during the course of instruction – of capturing the dynamic interplay between testing and teaching, as it goes on in human directed instruction. As emphasized in my monograph, highly adaptive instruction can dramatically reduce the amount of testing and the amount of instruction necessary to achieve mastery. To the extent that meeting individual needs in a timely fashion is important, this is a serious and fundamental limitation (cf. Scandura, 1964). Adaptive instruction is a cornerstone in both ITS and SLT-based instruction, the major differences being how, and the ease with which this can all be accomplished.

Clearly, I also agree with Paquette that higher order knowledge plays or should play an essential role in education. Paquette's argument that higher order knowledge cannot be derived from problem domains, however, is contradicted by our long history of research on SA. William Roughead, a student of mine back in 1968, demonstrated that "What is learned" in mathematical discovery, can both be identified and taught explicitly. Far transfer has been demonstrated in studies (e.g., cited in Scandura, 2007, pp. 173–5; 251) ranging from highly rigorous experiments (Scandura, 1974) to an entire book for teaching mathematics to teachers (1971), with stops in between covering such things as compass and straight edge construction in geometry (Scandura et al, 1974), algebraic proofs (Durnin & Scandura, 1977) and Piagetian Conservation (Scandura & Scandura, 1980).

Despite his use of formal representations, Paquette's views on higher order knowledge are quite different than my own – of a similar genre although not as extreme as those implicit in Socrates questioning, or teaching "mental discipline" through geometry in the early 1900s. The use of generic terms like "meta-knowledge" tend to camouflage understanding, making comparison difficult. One has only to recall Polya's (1960) important work on the role of heuristics in mathematical problem solving. Indeed, I had the pleasure of spending hours with Polya back in the 1960s, trying with limited success to convince the old master that one might identify heuristics more precisely.

It is not that heuristics, or "meta-knowledge" generally, cannot be identified or taught. The problem, as in the examples I presented today, is that higher order rules necessarily have limited domains of applicability. SA was partially systematic from its inception. Originally, however, we had no way to explicitly identify where a given higher order rule might or might not apply. SA in its present form gives explicit attention to defining structures representing where (higher as well as lower order) rules are applicable and where not.

Paquette's comparison of UCM with the execution of meta process(es) is also misleading. What Paquette calls generic skills are simply informal descriptions that can be made arbitrarily precise via SA – as higher order SLT rules. In contrast to meta-processes, UCM is a very precise, universally applicable and automatable executive process – one that controls the use of any and all higher (as well as lower) order knowledge. Formulating UCM in its current form required a series of insights to get right. UCM makes it possible to replace higher order SLT rules (representing generic skills et al) in a completely modular fashion. This means that it is now possible to build intelligent systems in which completely modular higher, as well as lower, order SLT rules can be modified without changing the way the cognitive system operates. This is not possible in expert systems, for

