A Deterministic AI Foundation for Modeling Human Tutors: Fundamental Assumptions in Structural Learning Theory

JOSEPH M. SCANDURA

Director of Research, TutorITweb.com, Emeritus Professor, University of Pennsylvania, USA, e-mail: scandura@scandura.com

This paper summarizes key stages in development of the Structural Learning Theory (SLT) and explains how and why it is now possible to model human tutors in a highly efficient manner. The paper focuses on evolution of the SLT, a deterministic theory of teaching and learning, on which AuthorIT authoring and TutorIT delivery systems have been built. It explains how SLT differs fundamentally from other theories used to motivate adaptive tutor development and how AuthorIT and TutorIT technologies differ from others used in developing adaptive learning systems. Implicitly, the paper also makes clear why it has been possible using AuthorIT to develop so many TutorIT tutorials in record time at minimal cost.

Keywords: Tructural Learning Theory, Structural Learning, Structural Analysis, What is learned, Intelligent Tutoring Systems, teaching and learning, Bloom’s 2-Sigma, Universal Control Mechanism, AuthorIT, TutorIT, GIFT, TICL, knowledge representation, ACT-R, BIG DATA, deterministic theory, higher order knowledge, short term memory

The ideas in this paper were first presented as a keynote address at the 2016 American Education Research Association (AERA), Technology, Instruction, Cognition and Learning Special Interest Group (TICL SIG) in San Antonio, TX. My keynote immediately followed a TICL symposium featuring adaptive learning technology talks by Bob Sottilare, John Dexter Fletcher and me. Bob’s paper

*Corresponding author: scandura@scandura.com
presented a broad framework for adaptive learning with eMAPS, serving as the current default adaptive learning engine for the Generalized Intelligent Framework for Tutoring’s (GIFT’s). Dexter’s talk presented a broad overview of research in the field, including tutor decision making in a DARPA funded project. Both papers build on theory that is inherently stochastic in nature.

My paper at AERA summarized the current status of AuthorIT and TutorIT (Scandura, 2016, 2017) along with historical developments leading to my Structural Learning Theory (SLT) (see Scandura & Novak, 2017). It concluded with a brief summary of relationships showing how AuthorIT and TutorIT offer an alternative approach to eMAPS as a means of achieving GIFT goals. Please see my first paper in this issue for an overview, references and the current status of AuthorIT and TutorIT with respect to goals set for GIFT (e.g., Sottilare, this issue). This and referenced papers describe our fully functional AuthorIT authoring and TutorIT delivery platforms.¹ Included are examples and summary lists of a broad sample of TutorIT tutorials as well as the AuthorIT authoring platforms used and refined in the development process.

This paper addresses three foundational concerns that motivated development of the SLT, and ultimately AuthorIT and TutorIT. The following ALL play an essential role in building any truly adaptive learning system:

1. The way knowledge is represented
2. How students solve problems
3. Short and long term memory.

These concerns also play a central role in most theories of learning. The Structural Learning Theory (SLT) starts from a larger perspective. Most essential was looking at the problem from the standpoint of teaching and learning as a whole. The first iteration of SLT was presented as a unified theory at an invitational conference at Penn on “Structural Learning” in 1970, a few days later at the Structural Learning SIG at AERA in 1970 and published in Scandura (1971).²

¹ While fully functional, these technologies like any complex software system are subject to further testing and refinement. Prospective partners and investors are most welcome. Contact us at Scandura@TutorITweb.com

² I first introduced the term “psycho-mathematics” in a paper “Research in Psycho-mathematics” published in the Mathematics Teacher (official journal of the National Council of Teachers of Mathematics (Scandura, 1968b). In the same time frame, Z. P. Dienes liked the name and shortly thereafter created an Institute for Research in Psycho-Mathematics at Sherbrook University in Canada. These initiatives and a National Council of Teachers of Mathematics (NCTM) monograph I edited in the same time frame led to NCTM establishing the Journal of Research for Mathematics, formally
Subsequent refinements and basic research supporting SLT were published over a period of years. Most important were Scandura (e.g., 1973, 1977, 2001, 2007). This article focuses on theoretical issues in SLT that pertain most directly to AuthorIT and TutorIT.

We will turn to the SLT in a moment but let’s begin with a short history of Intelligent Tutoring Systems (ITS). Brown and Burton (1977) built the first ITS, using theoretical principles first published in 1971 (Scandura, 1971; Scandura & Durnin, 1971; Durnin & Scandura, 1973). This work was followed by Anderson’s ACT-R theory and its use in building ITS (e.g., Anderson, 1988, 1995; also see Wheeler reference). Most of these ITS present students (Ss) with problems and attempt to guide them as they find their way to solution. Although well-funded for many years, neither ITS, nor contemporary successors based on BIG DATA (e.g., Knewton) focus on modeling the processes used by good human tutors.

Before moving to our work, let’s first consider Anderson’s ACT-R theory in a bit more detail. I choose ACT-R for three reasons. It has a) theoretical rigor and b) a long history.3 c) It also has motivated development of large number of ITS.

establishing a research foundation for mathematics education. Ironically, the term “Psycho-mathematics” is most associated with Dienes and “Structural Learning” with, even though Dienes was the one who founded the “Journal of Structural Learning”.

3 Our work has an even longer history. It began with detailed tape recorded analyses of the teaching – learning process using artificial content designed to parallel mathematics learning while minimizing the effects of prior learning (Scandura, 1964a,b). These studies were followed by a series of formal experimental studies using these same abstract materials (Scandura, 1966a,b,c). These were followed by studies involving mathematics learning (e.g., Scandura & Behr, 1966; Scandura & Roughhead, 1967; Scandura & Wells, 1967; Scandura, 1967a,b; Scandura, Woodward & Lee, 1967; Scandura & Wells, 1967; Scandura, 1967).

In parallel, we conducted a series of experiments in experimental and mathematical psychology (e.g., Greeno & Scandura, 1966, Scandura, 1967c,d; Scandura & Voorhies, 1971) along with theoretical analyses challenging S-R thinking which was prevalent at the time (e.g., Scandura, 1968, 1969a,b). Maverick that I was, I simultaneously was challenging prevailing thinking in mathematics education at the time (e.g., Scandura, 1968a,b, 1969; Roughhead & Scandura, 1968;Scandura & Durnin, 1968; Scandura & Anderson, 1968). This work provided an empirical as well as conceptual foundation (e.g., Scandura, 1968; 1969a,b; 1970; 1971a,b) for the Structural Learning Theory (SLT). SLT was first presented as an integrated theory at a Structural Learning conference at Penn in 1970 (see Scandura, 1971).

This article attracted a lot of attention and became an ISI Citation Classic (e.g., Scandura, 1987). It led to a long series of studies and theoretical extensions and refinement aimed at different, overlapping audiences, all in the same time frame. In retrospect, this made it increasingly hard for outsiders with their own research agendas to follow. Undoubtedly more important, the structural learning program at Penn with up to 20 full time PhD students ceased to exist after the mid-1970s leaving few able to carry on that line of work.

I personally turned my attention during this period primarily to software engineering as well as computer based instruction. It began with development of a wide range of Apple II tutorials in the late 70s & early 1980s, beginning with the first SAT prep tutorials, and progressing to fundamental
I also single out ACT-R because it makes theoretical assumptions that are at odds, in many ways opposite those on which the Structural Learning Theory (SLT) is based (e.g., Scandura, 1971, 1973, 1977, 2001, 2007). Just as ACT-R guided development of many ITS, SLT provided the foundation on which AuthorIT and TutorIT have been built and the way it is tested. In short, it offers a good counterpoint.

According to Anderson ACT-R theory (1995) complex cognition arises from an interaction of procedural and declarative knowledge in working memory. Procedural knowledge is represented in terms of production rules, and declarative knowledge in terms of “chunks”. A large number of chunks and transformations (productions) are assumed to underlie human cognition. From this large database, appropriate units are selected for use in particular contexts by an activation process tuned to the statistical nature of the theory. According to ACT-R, the power of human cognition depends on the amount of knowledge encoded and deployment of that encoded knowledge.

Wheeler presents a high level overview of the cognitive architecture underlying Anderson’s ACT-R theory (see referenced link). This overview shows a sharp distinction between Declarative and Procedural knowledge in working memory. This distinction is fundamental not only in ACT but shared by most researchers in the ITS community.

Accordingly, ACT-R

1. Represents knowledge in terms of chunks and production rules.
2. Students solve problems by selecting these chunks and transformations (productions).
3. Working memory operates on both declarative and procedural knowledge as summarized in Figure 1.

In short, traditional ITS represent to-be-learned knowledge in terms of chunks of knowledge and production systems with usage controlled by various assumed control mechanisms. In ITS, students (Ss) are allowed to solve problems as they wish with guidance offered along the way.

issues in software engineering (e.g., Scandura, 1991, 1994a,b). Serendipitously, this led to solutions to some of the outstanding limitations of SLT enabling development of the AuthorIT authoring and TutorIT tutoring systems as they exist today. The SLT rests on broad and deep theoretical foundation, parts of which remain to implemented in AuthorIT or TutorIT.

For those so motivated much of this work has been categorized in an appendix to Scandura & Novak (2017).
ACT-R LIMITATIONS

A major problem any ITS developer must face is the level of (knowledge) representation chosen. This depends heavily on how advanced the student population, and even more important on the range of abilities in the S population.

More fundamentally, ACT-R is inherently a stochastic theory. Chunks and productions lie in large data bases and are selected in accordance with a variety of mechanisms assumed to control their use. In ACT-R the processes involved are inherently stochastic in nature. Presumably, productions and chunks hang around in working memory and are called upon as dictated by various stochastic assumptions. Generally speaking, ACT-R builds tutoring systems from the bottom up. Complex behavior is the result of combining productions and/or chunks.⁴

The situation is further complicated because experience has shown that it is difficult to identify ALL things Ss might do. Hence, ITS invariably focus on a small number of options. Moreover, the focus in most ITS has typically been limited to procedural knowledge. Commercial curricula based on ITS appear to

⁴ Conversely, as we shall, see SLT works from the top-down (Scandura, 2013a).
have left declarative knowledge for the future – consider the reams and reams of workbook pages used for years in Carnegie Learning’s math curricula.

Despite attempts to speed the process, ITS are still time consuming and expensive to build and adjust (e.g., Murray, 1999; Gilbert et al, 2011). Moreover, ITS have been most successful with high performing students. These are the Ss who traditionally need the least help. Given high development costs and limited success with underperforming Ss, it is not surprising in that ITS generally are either undergoing major modification to speed ITS development and/or are being discarded in favor of BIG DATA.

**BIG DATA LIMITATIONS**

Adaptive learning systems based on BIG DATA take stochastic decision making to a new level. Adaptive learning systems based on BIG DATA focus on how students behave on average (e.g., Scandura, 2016). If S gets a problem correct, he or she is given a problem Ss on average find more difficult – and vice versa on failure. Although other factors often play a role, decision making using BIG DATA is based on automated statistical analysis of large amounts of data in the hope this will be sufficient. Human judgement plays at best an indirect role in the process (for further discussion, please see Scandura, 2017).

**DETERMINISTIC PRINCIPLES GUIDING SLT DEVELOPMENT**

No attempt is made here to detail the approach based on SLT with those motivating the other approaches. This has been done elsewhere. Interested readers are encouraged to read recent theoretical notes and papers comparing SLT with ITS and BIG DATA (Scandura, 2013a, 2014, 2016).

The ideas expressed in this paper were first presented after talks by Sottilare on GIFT integrating a variety of technologies (a successor of ITS and beyond), by Fletcher originally on BIG DATA herein focused on DARPA’s Digital Tutor and by me on AuthorIT and TutorIT (e.g., Scandura, 2016, 2017; Scandura & Novak, 2017).

Looking back, SLT has generally been correct from its inceptions (e.g., Scandura, 1970, 1971, 1973, 1974, 1977). Our original research was conducted under idealized conditions, providing deterministic support for basic assumptions in SLT. That is, it was possible to predict the behavior of individual Ss in specific situations with deterministic precision. The limited deviations found could easily
be attributed to factors extraneous to these basic assumptions – like ignoring friction in testing basic laws of classical physics – e.g., moving objects up an inclined plane.

Given a problem domain, considerable attention was given from SLT’s inceptions to identifying higher as well as lower order knowledge (SLT rules) necessary and/or sufficient for success in relatively large problem domains. Specifically, the introduction of higher as well as lower order SLT rules made it possible to account for problem solving in complex as well as simpler domains. As above, the method of Structural Analysis (SA) has been widely used to systematically identify both lower and higher order SLT rules in a variety of domains (e.g., Scandura, 1977). These studies included straight edge and compass constructions in geometry (Scandura, Durnin & Wulfeck, 1974) and proving theorems in algebra (Scandura & Durnin, 1977). Continuing refinement of SA came later (e.g., Scandura, 2001, 2007).

This work built on a broad base of earlier research conducted with the help of many former students and colleagues. A large number of experiments on rule learning and problem solving were conducted during the later 1960s and 1970s. As I look back, perhaps the single most important conclusion of this research was that what a student learned or had to learn was far more important than how (cf. Roughhead & Scandura, 1968 and Scandura, 2014). Decades of research confirmed that the results of any experiment involving complex content are predictable with deterministic precision to the extent that one can specify what needs to be learned for success AND that the student actually learns what has been specified. Affective and/or other factors, of course, might reduce the likelihood of student learning, or require more time. What matters fundamentally, however, is whether the learning in fact takes place.

As above, motivated by the determinism characterizing classical physics, I adopted the goal early on of seeking theory that made it possible to predict the behavior of individual students in particular situations. No one else was doing this at the time (or since). Hence, you might ask, “How is this possible?” All I can say is “It was”, and unless the laws of nature have changed “It is”.

Appreciating the importance of this fact requires a fundamental change in perspective. Sophisticated psychometric techniques and methods of analysis may make one feel more secure, but all they do is make it more difficult to separate wheat from the chaff.

---

5 See Scandura, J. M., Koedinger, K, Mitrovic, T, Ohlsson, S. & Paquette, G. (2007) for the only publication including scientists from both ITS and SLT perspectives. This commentary was based on articles published in a special issue of TICL in 2007.
This line of research has a long history. It began with an array of studies ranging from detailed analysis of tape recordings of teacher-student interactions (Scandura, 1964a, b). These were followed by formal experiments in both classroom and laboratory settings. Dozens of research studies showed that experimental results are predictable to the extent that one can specify ahead of time what needs to be learned for success.

Given these results, it seemed axiomatic that the focus in tutoring systems should be on ensuring that students learn what is needed for success. As in early physics, our core experiments were run under “idealized” conditions. We eliminated unwanted factors to focus on the variables being studied. Imagine Galileo running his famous experiment at the Leaning Tower of Pisa. He dropped two iron balls of different weights. Surprisingly (at the time), they reached the ground at the same time. For fun, the author replicated this result in the 1970s before they closed the open tower to visitors. But I added a wrinkle. Instead of a large stone and a small stone, I dropped a small stone and a feather.

Why didn’t Galileo do this? If he had, instead of classical deterministic theory, physics might instead have evolved very differently – into a theory of droppings with each type of object falling at its own rate. In effect, what Galileo did was ask what would happen under idealized conditions. What would happen if there were no air resistance?

SLT was motivated by the same kind of thinking. As above, my first attempt to put it all together was called “Deterministic Theorizing in Structural Learning” (1971). SLT was further developed and expanded in my book, Structural Learning I: Theory and Research in 1973, which is currently being reprinted by Taylor & Francis. This book was followed in 1977 by Problem Solving: A Structural/Process Approach with Instructional Implications. Part 1 in this book included a detailed update of the SLT and a broad spectrum of empirical studies. Part 2 dealt with Structural Analysis (SA) applied to compass and straight edge construction problems and algebraic proofs. Part 3 included cognitive mechanisms including the role of goal switching (UCM) in a broad range of problem solving, rule retrieval, problem definition and rule selection situations. Part 4 on

---

6 The approach I’ve taken is best understood in comparison with the approach taken by two of my early colleagues. Pat Suppes (1967), a logician turned psychologist and pioneer in CBI, was motivated by rigor, believing science progresses best when hypotheses can readily be proven wrong and corrected. Zed (Z. P.) Dienes a mathematician also immersed in making mathematics assessable to young children took the opposite position (1963). Dienes (personal communication) said he would “rather be vaguely right than precisely wrong”. I add these side comments because they helped motivate my approach to the subject. I wanted to be correct – but also as rigorous as possible. I might add in retrospect – even if it took several decades to achieve that goal.
individual differences updated work on assessing behavior potential – finding out what individual students did and did not know. Part 5 dealt in turn with instructional applications.

In retrospect, I had and am grateful to a large team of talented PhD students at the time. Without them, this work would never have been possible. On the downside, findings and reports were being produced at a rate that few outsiders had the time to keep up with. Overly complex theoretical writings at the time (for which I am responsible) didn’t help.

The latest more or less complete formulation of SLT was in the TICL journal (Scandura, 2007). This paper confirmed and updated the following.

1. Ignoring time and memory limitations what would be necessary to enable a person to solve a given class of problems? How can one represent what is to be learned? That is, “what” must be learned for success under idealized conditions – ignoring how, much as in Chomsky(ian) linguistics.
2. What does a person need to know to solve a given class of problems, or to acquire new knowledge? How might learning and behavior take place under idealized conditions (i.e., with perfect memory and unlimited time).
3. How is behavior affected by each learner’s processing capacity and processing speed. In the first case, think of Miller’s magic number 7+/–2 (Miller, 1956). Rather than viewing 7+/–2 as an average, Scandura (1971, 1973) and Voorhees and Scandura (1977) found that some individuals were five “chunkers”, others eight, etc. In short, our data showed that each individual has a fixed capacity for processing information – irrespective of the task involved. Another assumption made in SLT is a characteristic processing speed. That is, some people are assumed to characteristically process information more quickly than others. This remains a commonly made observation seeking hard deterministic data.8

The SLT is fundamentally different in these respects from ACT-R or for that matter any other behavioral theory I am familiar with. Whereas ACT-R like other cognitive theories rests on a stochastic foundation, SLT is inherently deterministic. It readily and directly lends itself to (deterministic) software implementation.

---

7 “what” in this context refers to learning as a construct, irrespective of the conditions under which this learning takes place.

8 It is important to stress that the required data is quite different than that obtained in most behavioral research. In testing, it is essential that Ss process information in predetermined ways. This requires careful pre-training under carefully controlled conditions (e.g., see Voorhies & Scandura, 1971, 1973, 1977).
With training in mathematics education and mathematical foundations as well as
statistics, I early on rejected the idea of stochastic theorizing as the best way to
understand the teaching-learning process. With the sheer beauty of deterministic
Newtonian physics as a guide, I asked why we couldn’t do the same in behavioral
science, or at least in studying and understanding the teaching – learning process.
Does that mean we can (or could) predict human behavior with exactitude and,
with teaching as a goal, fully control human behavior? No, but then neither can
one predict the exact force needed to move an object up an inclined plane, or the
exact rate at which a real object will fall on earth. Friction and air resistance
inevitably take their toll in our real world.

Fortunately, Galileo and Newton focused on essentials. Neither was able to
reproduce observable behavior with theoretical certitude. Still, they did not resort
to stochastic measures. If they had, instead of the sheer beauty of Newtonian
Physics, we might well as above have ended up with a stochastic science of drop-
pings. What early physicists were able to do was to say (and predict) how things
would behave under idealized boundary conditions – in situations where there is
no friction or air resistance.

In my initial formulation of the Structural Learning theory (SLT) I asked myself
the same questions. Given some domain of behavior, why couldn’t one ask what
needed to be learned to produce that behavior? And why couldn’t we predict what
a given human would be to do if we knew (or assumed) what he knew, and he had
all the support necessary to employ his knowledge. That is, what if we removed real
world constraints – so humans had all the time and memory resources they might
need. In short, what would behavior be like under idealized boundary conditions?

The SLT is inherently deterministic in nature. This was reflected in the title of
my first article on the subject: “Deterministic Theorizing in Structural Learning:
Three Levels of Empiricism”. Given a problem domain:

1. The first thing a teacher or tutor needs to know is what Ss need to know for
   success in the given domain. Notice that this is at odds with the notion of
   allowing Ss to proceed as they will – with an ITS offering guidance along the
   way.
2. Second, how is that knowledge put to use in solving problems in the domain
   of interest? How would a S behave under idealized conditions – given suffi-
   cient time and perfect memory.
3. Third, how is behavior affected when memory is taken into account? In the
   absence of memory aids, behavior is constrained by limits on the learner’s
   ability to process information – think Miller’s magic number seven plus or
In short, rather than trying to predict average behavior, SLT is designed to predict the behavior of individual students in particular situations — under idealized conditions.\(^9\)

The goal of SLT from its inceptions (Scandura, 1971) has been to provide a comprehensive, rigorous, precise, operational and now computational theory of teaching and learning. It offers a rigorous account of the way tutors (and teachers) interact with their students. Data confirmed that the knowledge associated with arbitrarily complex task/problem domains can be represented in terms of SLT rules and higher order rules (which operate on other SLT rules to generate new SLT rules).\(^{10}\)

This initial form of SLT offered a precise account of individual problem solving under idealized, memory free conditions. Again, by idealized conditions, I mean human behavior collected in situations where the effects of memory and processing time are minimized if not eliminated. Accordingly, we were able to pinpoint what each S would do in given problem solving situations under pre-specified “memory-free” conditions. Fundamental assumptions in SLT were confirmed in a series of experiments conducted under idealized conditions (e.g., Scandura, 1970, 1971, 1973, 1974, 1977; Durnin & Scandura, 1973, Ehrenpreis & Scandura, 1974, Lowerre & Scandura, 1973, 1974).

A study conducted with Don Voorhies took Miller’s (1956) classic 7 plus or minus 2 results in short term memory a step further. Data (Scandura, 1971; Voorhies & Scandura, 1977) demonstrated that each individual’s behavior is constrained by that person’s (fixed) capacity for processing information — irrespective of the task involved. Based on informal observation, SLT also hypothesizes that each individual has a characteristic processing speed (1971, 1973, Chapter 7 in Scandura, 1977).

---

\(^{9}\) Again, don’t be misled by the myriad of factors that may stand in the way of success by any particular learner. Analysis of any given content domain begins with basic assumptions as to minimal capabilities. This is inherent in the process of Structural Analysis (SA). SA results in the hierarchical knowledge representations needed to begin the process. In teaching column subtraction, for example, we begin with the assumption that every student in the target population begins knowing the basic facts. One can always assume less, of course, but then the required Structural analysis (SA) must accordingly be extended.

\(^{10}\) The original form of SLT built on years of work beginning during my undergraduate days at Michigan. The mathematician George Polya’s “How to Solve It” (1945, Princeton Univ. Press) was among the first attracting my attention. Toward the end of his career, I had the honor of his devoting an entire day with me as I tried to explain my early work analyzing the teaching learning process (Scandura, 1964a,b). I found that when information was given relative to what the student knew at the time played critical role in student understanding. After listening carefully to my explanations, he replied each time with the equivalent of, “I don’t understand”. Buildings built on slush don’t last long. What he was trying to do was to get me fill in the blanks – gaps in my thinking. As Einstein once said “The whole of science is nothing more than a refinement of everyday thinking.”
Aside from processing speed, all fundamental assumptions of SLT have been subjected to and confirmed in both deterministic testing with individuals and in group testing (e.g., Scandura, 1970, 1971, 1973, 1974, 1977; Scandura & Durnin, 1971/1978; Durnin & Scandura, 1973, Ehrenpreis & Scandura, 1974; Lowerre & Scandura, 1973; Scandura, Durnin & Wulfeck, 1974; Wulfeck & Scandura, 1977).

To summarize, basic assumptions in the initial form of SLT were as follows:

1. To-be-acquired knowledge associated with any given task or problem domain can be represented as SLT rules and higher order SLT rules. SLT rules were originally represented as directed graphs (think flow charts consisting of operations and decisions). Each SLT rule required an assumed level of representation.
2. A deterministic account of any subjects (S's) problem solving behavior involving higher as well as lower order knowledge (SLT rules) is possible when testing is conducted under idealized (memory free) conditions. Higher order SLT rules are used to derive needed lower order ones as needed. (NOTE: While detailed experimental confirmation is limited, recovery from long term memory is assumed to take place in the same manner.)
3. Each S has a fixed capacity for processing information and a characteristic processing speed. Introduction of these constraints will have predictable effects on S behavior on specific tasks under specific conditions.

INITIAL FORM OF SLT

The major question we faced in building AuthorIT and TutorIT was how to implement a deterministic system that meets the above requirements. Rather than invoking or building on theory that is inherently stochastic in nature (e.g., as in ACT-R), deviations of observable from predicted behavior were as in classical physics attributed to errors of measurement. My initial formulation of SLT (Scandura, 1971) was a first important step in this direction.

1. Given a content domain, I showed how well defined behavior could be explained in terms of what we later called SLT rules and higher order SLT rules (e.g., Scandura, 1971, 1973). Higher order SLT rules in this case

NOTE: We have used the term “SLT rules” to distinguish them from “production rules” originally used in mathematical logic, later adopted by Newell & Simon (1972) and still later by Anderson in
operated on given SLT rules to generate new ones. See below for initial limitations and subsequent advances used in developing our AuthorIT and TutorIT platforms. It’s essential to note that higher order rules are fundamentally different from control mechanisms commonly used in ITS. The latter are fixed assumptions in the mind of the theorist. The higher order rules I have in mind are derived from the domains in question.

We also needed a control mechanism to make things work – but let’s not get ahead of ourselves. We will return to that later. Suffice it to note here that higher order SLT rules are derived via an increasingly systematic process called Structural Analysis (SA) (e.g., Scandura, 1977; Scandura, Durnin & Wulfeck, 1974; Scandura & Durnin, 1977).

2. A second step in this direction involved explaining and predicting individual behavior in problem solving situations under idealized conditions – given sufficient time and perfect memory. That is, we were concerned with predicting behavior in situations where Subjects knew (i.e., had access to) both the lower and/or higher order knowledge needed and all the time necessary.

In SLT, this required introduction of what I called a Universal Control Mechanism (UCM) for controlling the use of needed SLT rules. As we shall see below, formulating UCM in a way that was both independent of any higher order SLT rule and executable on a computer was far from obvious and came considerably later.

3. The deepest level of SLT also considered how behavior is affected when memory and processing speed are taken into account. Although first introduced in the same time frame (e.g., Scandura, 1971), implications of working memory with a limited capacity for processing information and processing speed is the least well developed, only partially tested part of SLT. It has been only indirectly implemented in TutorIT.

LIMITATIONS AND PROBLEMS WITH THE INITIAL FORM OF SLT

Although intuitively satisfying, none of the (above) assumptions on which SLT is based readily lent themselves to implementation.

1. We didn’t have a systematic, repeatable way to define the process of Structural Analysis (SA). Such an SA process would be essential to identify the

ACT-R. For lack of a better formalization, we originally used Flow Charts and (higher order) directed graphs operating on other directed graphs for our purposes.
SLT rules and higher order SLT rules necessary for success in any given problem domain. SLT rules identified via SA consist of component operations and decisions (rather than simple lists of individual productions). In another sense, however, SLT rules originally shared a fundamental problem with production systems. What is the appropriate level of detail one should use in representing the knowledge associated any given domain?

2. Second, we couldn’t define UCM in a way that was truly independent of the higher order rules themselves. Consequently, our UCM was fundamentally no different from the various control mechanisms (e.g., means-ends analysis, chaining, generalization, analogy, etc.) assumed in various ACT-R theories. No gain there. Again, what to do?

3. Our main contribution with respect to working memory and processing speed was finding a way to determine memory load in solving given tasks in pre-determined ways. Memory load, for example, depends heavily on the solution methods used, external memory aids, etc.

Determining an individual’s processing capacity requires exacting pre-training to ensure that Ss performed tasks exactly as prescribed. Otherwise, memory loads would necessarily differ. TutorIT currently provides unprecedented adaptivity based on what any given S knows at any given time. TutorIT does not yet, however, make explicit provision for differences in individual memory capacity or processing speed.

Each of these issues/concerns is fundamental and has far reaching implications. Let’s consider each in turn.

1. How to Represent Knowledge? With these concerns still in the back of my mind, most of my attention after the late 70s moved to software engineering. Oddly enough, our work in software engineering helped shape solutions to the most pervasive problems. How can one represent, even know what students need to learn to be successful in any given problem domain? Specifically, what is the appropriate level of detail in representing knowledge?

The short answer to the above question is that there is no one level that works best when talking about populations of students. Nor, as we shall see below (e.g., Figure 4), it is not necessary or desirable in most cases to settle on only one level of representation. Needed was a way to represent ALL levels of refinement (i.e., of knowledge representation) – simultaneously.

2. How to represent UCM in a way that is independent of higher order SLT rules? The problem of distinguishing UCM from other higher order knowledge (SLT rules) was subtle and remained for some years. Evaluating S responses requires knowing or being able to generate answers to given problems. This
requires some way to either combine and/or modify available knowledge however it is represented, whether in the form of productions or SLT rules.

This is accomplished in ITS by introducing control mechanisms (e.g., chaining, analogy) deemed suitable for the domain in question. SLT demands more. **We needed a control mechanism that works independently of the domain in question.** But, it took some time, and eventual insight to find one. Without such a UCM, SLT and hence TutorIT and AuthorIT would be no different in this respect from ITS (using a variety of control mechanisms in various domains.)

**OK! LET'S GET DOWN TO SOLUTIONS**

SLT is a deterministic theory that aims to predict the behavior of individual students in specific situations. You might ask “How is this possible?” Again, SLT is designed (using principles common in classical Newtonian physics) to explain behavior under idealized conditions (cf. Scandura, 1971, 1973, 1987, 2001, 2007).

To provide answers, let me again summarize how the SLT evolved, which parts have been implemented in AuthorIT and TutorIT and which parts are either secondary, or remain for the future. The essential result of this work is a theoretical framework that models the human tutoring process as a whole – with arbitrary degrees of precision.

SLT begins with a fundamentally different assumption than other formal theories employed in building adaptive learning systems. For one thing, Declarative and Procedural knowledge are just different views on a single knowledge representation (e.g., Scandura, 2016, 2017). For another, explicit provision in SLT is made for higher order knowledge (cf., Scandura, 1971, 1973, 2001, 2007).

AuthorIT and TutorIT build directly on SLT. The focus is on WHAT needs to be learned for success. HOW instruction is provided is of secondary importance. My goal here is to summarize and clarify the theoretical assumptions underlying the tutoring process.

---

12 Whereas many attempt to automate the entire tutoring process (e.g., using BIG DATA or ACT-R), our work is based on the assumption that tutoring optimally requires both human judgment and automation (cf. Time Magazine article on AI pioneer David Gelernter, March 7, 2016). My goal here is to focus on fundamental differences in the theoretical foundations on which AuthorIT and TutorIT are based that offer solutions to the above problems. AuthorIT and TutorIT are definitely NOT eclectic in nature. As above, these technologies build directly on the Structural Learning Theory (SLT), a comprehensive, operationally defined deterministic theory of complex human learning AND teaching.
As summarized in Figure 2, SLT provides a broad framework for both explaining and predicting the behavior of individual students under idealized conditions. The remainder of this article focuses on how SLT has evolved over several decades based on testing under both laboratory and practice conditions, technical developments based on SLTs use in software engineering and their application in developing AuthorIT and TutorIT software. Why use SLT as a foundation for building tutoring systems? While sometimes sharing overlapping concepts and terminology with ITS (e.g., Anderson, 1988; Ritter, 2005), the goal of SLT from its inceptions has been different. Most (all) cognitive theories have focused first and foremost on how students learn. The process of teaching has traditionally been an add-on. SLT is different.

From inceptions, SLT has focused on the teaching-learning process as a whole. It focuses first and foremost on what student must learn for success. Together with technical advances, key assumptions in SLT have become increasingly precise. This evolution over a period of decades made it possible to model the human tutoring process. As above, there is a big difference between modeling
human tutoring as an integrated process versus adding tutoring on top of a learning theory (e.g., as in ACT-R or ALEKS), or for using BIG DATA in an attempt to automate the entire process.

Cognitive theories have focused on how students learn (see above). Teaching and tutoring have been add-ons. In contrast, SLT focuses on the teaching-learning process as a whole. FIRST, a good human tutor must know the subject matter to be taught. Minimally, good tutors must know the subject matter, what anyone including students need to know to be successful. At minimum, they must be able to distinguish acceptable from the unacceptable. Traditional ITSs assume the primary role of tutors is to keep Ss from making mistakes. Indeed, some ITSs allow students to make mistakes so they can encourage them to correct those errors. SECOND, a good human tutor must know how to determine what a given student does and does not know at each point in time with respect to what is to be learned. THIRD, a good human tutor must have at least an intuitive understanding about how students use what they already know to acquire new knowledge that may be needed. FOURTH, knowing what is to be learned, what any given student does and does not know at each point in time and how students learn (which is where higher order SLT rules come in), a good tutor must be able to put it all together.

In short, good human tutors must know: a) what must be learned for success, b) how to determine what any given student does and does not know at each point in time, c) how Ss use what they know to acquire new knowledge and d) what each S needs to progress at each point in time. Ideally, the tutoring process should continue until the student demonstrates some predetermined or otherwise acceptable level of mastery.13

AuthorIT and TutorIT build on this foundation. Implemented in AuthorIT, Structural Analysis (SA) is a relatively easy to learn. SA is a largely automated method enabling Authors/Instructional Designers/Subject Matter Experts (SME) to represent what is to be learned for success. To-be-learned knowledge in SLT is represented hierarchically – simultaneously representing ALL levels of expertise.

Structural Analysis makes it possible to systematically identify the SLT “rules” and “higher order rules” collectively necessary and sufficient to represent to be

---

13 Terminal elements in any complete Structural Analysis (SA) of to be learned content correspond to prerequisites assumed to have been mastered by students prior to entry. Achieving this level of precision in SA is possible with many subject matters, basic mathematics being one of them. Analysis of “fuzzier” content, such as how to write good poems, generally will be far less complete. In such cases, it will be desirable to require success on more than one example to increase the likelihood of mastery.
acquired knowledge associated with arbitrarily complex domains. Examples include published analysis of straight edge and compass constructions in geometry (Scandura, Durnin & Wulfeck, 1974, Wulfeck & Scandura, 1977) to basic algebra (Durnin & Scandura, 1977) and Piagetian conservation (Scandura & Scandura, 1980).

Knowledge presentations (KRs) consisting of SLT rules represent what SMEs believe students must or should learn to be successful. SLT rules effectively serve as “rulers” for measuring what individual students know. They provide an explicit basis for determining what any given S does and does not know relative to what is to be learned, and what students need to progress at each point in time (e.g., Durnin & Scandura, 1973; Scandura & Durnin, 1978).

Structural Analysis (SA) is a highly systematic process that has been implemented in AuthorIT. Having said that, SA purposefully requires human judgement. I do not believe it either possible or desirable to try to fully automate this process since it intrinsically requires deep subject matter knowledge available by definition to experts in each domain area. Although not requiring pedagogical knowledge as such, it does require the ability to externalize that knowledge. SA is used to represent SLT rules that are both necessary and sufficient for success in whatever domain may be of interest. The goal is to make this process as fool proof and easy to use (by SMEs) as possible. There is a lot more to be said about this process of SA – see below.

Given SLT rules created via SA, TutorIT must make ALL decisions automatically. How might this be accomplished? Deterministic research (i.e., research predicting the behavior of experimental Ss in specific situations) has demonstrated that learning takes place when higher order rules operate on lower order rules (Ehrenpreis & Scandura 1974; Scandura, 1974; Scandura & Scandura, 1980).

SLT’s fundamentally simple (but initially only informally defined) Universal Control Mechanism (UCM) provided the glue controlling the way individual SLT rules (including higher order rules) interact. From inceptions, UCM was shown empirically to be sufficient to explain the way SLT rules interact as students solve problems (Scandura, 1971, 1973, 1974, 1977).

As mentioned above, two MAJOR PROBLEMS remained until recently. (NOTE: Another two theoretical assumptions are still largely missing from AuthorIT and TutorIT – see concluding remarks):

1. Given any problem domain, one must choose a useful level of detail in representing SLT rules. What is the best level of detail in representing SLT rules? The short answer is that this depends on the prior knowledge and
sophistication of the S population. This problem of identifying an appropriate level of knowledge representation is shared with all knowledge representation schemes (e.g., production systems). Given most S populations, the reason is that there generally is no one such level of representation.

2. How are we to represent the knowledge associated with large domains? UCM could NOT for many years be defined independently of higher order SLT rules associated with given domains. Until this fundamental issue was resolved, interaction among SLT rules would not be fundamentally different than control mechanisms used in ITS (e.g., means-ends analysis, analogy, etc.). The central question: Is it possible to represent (define) a Universal Control Mechanism (UCM) that is BOTH independent of any content domain and sufficient to explain, predict and control explicit behavior?

These limitations made it impossible to formalize and thereby fully implement tutoring systems based on SLT as executable (computer) programs. Rigorous theoretical solutions to these problems came as periodic insights, deriving largely from using Structural Analysis (SA) in SLT as a foundation for a cognitive approach to software engineering (e.g., Scandura, 1992, 1991, 1994a,b). In short, a decade long effort to develop a systematic cognitive approach to software engineering led to concrete answers to these two questions. The resulting methods have been patented (US Patent No. 8,750,782) and offer two unique benefits.14

WHAT IS AN APPROPRIATE LEVEL OF DETAIL?

The first fundamental problem is determining the appropriate level of detail for representing to-be-learned knowledge.15 Unlike ITS research, we rejected productions systems as allowing too many degrees of freedom, to my knowledge making it impossible to devise a universal mechanism for controlling behavior.

14 Our goal is make AuthorIT authoring and TutorIT delivery technologies as broadly available as possible – both for research purposes and to commercialization partners. AuthorIT and TutorIT will be of special interest to publishers, tutoring and training companies who want to distinguish themselves in the industry.

15 The problem of grain size is universal and even more complex in ITS. Cognitive Task Analysis is as much art as science (e.g., Koedinger, personal communication). Production systems, for example, work fine for representing procedural knowledge – but not declarative (I prefer structural) knowledge. As mentioned above, it is not surprising that Carnegie Learning practitioners were forced to add reams of Workbook pages to cover a standard algebra course.
Initially, we represented SLT rules in terms of directed graphs, or equivalently as flow charts.\textsuperscript{16} As with production systems, one can represent the knowledge necessary for solving any given class of tasks at any desired level of detail. \textbf{What, however, is that appropriate level?} This depends on the student population. What to do? Lacking a well-defined alternative, most instructional designers begin by assuming a minimal level of knowledge that entering students may be assumed to have.

The problem is that any given S may enter at any level – ranging from knowing nothing about the domain (other than basic prerequisites) to full mastery, or any place in between. There is no one ideal or preferred level of representation. A \textbf{complete knowledge representation must simultaneously represent ALL possible levels of mastery.}

\textbf{Solution – Represent SLT Rules Simultaneously at All Levels of Abstraction}

How to do this? Our work in software engineering indirectly led to an ideal solution. \textbf{Structured} analysis is a well-known method in software engineering for representing procedural knowledge at multiple levels of abstraction. Data analysis does the same for data (e.g., see Yourdon, 1975, 1979). Both, however, are essential in knowledge representation. Declarative (I prefer the term “structural”) knowledge involves what is being operated on at any given point in time. Procedural knowledge refers to actions on data.

In our work, we discovered that there is a close relationship between data analysis and procedural analysis. Furthermore, there is a simple direct correspondence between procedural and data refinements (e.g., Scandura, 2003, 2007).

Moreover, a very small number of data and corresponding procedural refinement types is sufficient (e.g., Scandura, 2003): Refinement of data elements into COMPONENTS corresponds to refinement of operations into subordinate SEQUENCE or PARALLEL operations. Similarly, CATEGORIES of data correspond to IF..THEN/CASE or REPEAT..UNTIL/DO..WHILE refinements in procedures. DYNAMIC data refinements require operations. They correspond to what we call INTERACTION refinements in procedures. The latter convert relationships (used in relational models) to equivalent procedures. (These parallels follow from a basic equivalence in mathematics between functions and relations. Each relationship effectively defines a set of functions.)

In short, \textbf{a small finite number of refinement types has been shown to be sufficient for representing any data structure and/or corresponding procedure}

\textsuperscript{16} Before this I had explored any number of mathematical representations (cf. Scandura, 1973). The closest I came to what we have now (Abstract Syntax Trees - ASTs) were directed graphs.
hierarchically. These hierarchies make it possible to represent knowledge at any and all relevant levels of expertise simultaneously (e.g., Scandura, 1994a, 1994b, US Patents 6,275,976 & 8,750,782, 2003). In short, given any well-defined content, it is possible irrespective of complexity to represent ALL content hierarchically – from the highest levels of abstraction to the lowest levels of detail (corresponding to prerequisites assumed for any given student population).

Representing knowledge hierarchically in terms of what we call Abstract Syntax Trees (ASTs) effectively solves the fundamental problem of choosing an appropriate level of detail. Any domain can be refined hierarchically to any desired level of detail. The minimal level in this case must/should match the level of expertise associated with the weakest students in any given target population. Representing knowledge hierarchically makes it possible to systematically identify and represent knowledge required for success in any well-defined domain at ALL levels of (potential) expertise – simultaneously. These ideas are summarized in Figure 3.

Figure 3 offers a solution to the problem of what is the appropriate level to represent knowledge. Rather than being forced to choose a FIXED level of expertise, or having to distinguish between two different kinds of knowledge (declarative and procedural), ALL knowledge is represented in terms of SLT rules (referred to as Abstract Syntax Trees [ASTs] in software engineering). Figure 3 lists a small number of refinement types shown to be sufficient for representing knowledge associated with any given problem domain. ASTs also make it possible to represent any and all levels of expertise.

In Figures 4A&B below, hierarchies are displayed sideways, from left to right. Higher levels in the procedural hierarchy (Figure 4A) correspond to operations on the more complex data structures shown to the left in Figure 4B. These operations are more automatic and hence correspond to higher levels of expertise. As we shall see, representing knowledge in terms of hierarchical ASTs also makes it possible to assess levels of expertise of individual students.

---

17 Since some readers may be steeped in instructional design (ID) (e.g., Gagne, 1985; Merrill, 1983), let me try to clarify the difference. A major focus in ID has been on how to teach different kinds of learning, specifically facts, concepts, procedures and variously called principles or problem solving. Knowledge representation in SLT requires only one distinction, SLT rules and SLT rules that operate on other (non-degenerate) SLT rules. The latter corresponds to problem solving in ID. Facts, concepts, procedures, on the other hand, correspond to various levels in SLT rule hierarchies. Like a fact, the top level involves directly generating a response to a given a problem or task. Some levels involve simple decisions, is it this or that – in this case like concepts in ID. Combinations of operations and decisions in an AST hierarchy (SLT rule) correspond to procedures in ID.
In effect, Declarative and Procedural knowledge are just different views at various levels of abstraction on a single knowledge representation. One corresponds to data. The other corresponds to procedures operating on that data.

Figure 4A shows how AST-based SLT rules make it possible to represent the level of expertise of any given student. (Think of this representation as a hierarchy turned on its side.) The highest level node, corresponding to full knowledge of the corresponding set of tasks is at the top. Lower level nodes correspond to knowledge associated with solving various parts of any given problem. In this case the green nodes are preceded by a plus (+) sign indicating that a student has demonstrated mastery of the corresponding skill. Black nodes preceded by a black question mark (?) represent knowledge whose status is still to be determined. Failed nodes preceded by a minus (−) sign would be red (not shown in Figure 4A).

Higher levels in a procedural hierarchy operate on higher levels in data hierarchies, and vice versa. Higher level operations in a procedural hierarchy operate on more complex data structures. In effect, automation of procedural knowledge
FIGURE 4A
Shows how SLT rules may be represented as procedural Abstract Syntax Tree (AST) hierarchies. SLT rules make it possible to simultaneously represent ALL levels of procedural expertise.

FIGURE 4B
Shows how the data structure on which the procedural hierarchy in Figure 4A operates also can be represented as an AST hierarchy.
involves higher level operations operating on more complex data structures. These higher level operations correspond to higher levels of expertise. Behavior is simpler and more direct. The knowledge is automated.

Conversely, lower level operations in a procedural hierarchy, operate on simpler data structures. Compare this with assumptions in ACT-R where automatization of procedural knowledge corresponds to compilation. In SLT, the process of automation corresponds to “chunking”, converting procedural knowledge to declarative (or structural knowledge). The results are the same in both theories – faster behavior. But, assumptions are reversed. In effect, declarative and procedural knowledge become just two aspects (data and process) of the same knowledge representation.

In short, there is no need to distinguish declarative (I prefer the term “structural”) and procedural knowledge. All knowledge in SLT necessarily involves both. Put differently, SLT rules represent ALL possible levels of expertise, from neophyte to master. SLT, in turn, makes it possible to determine precisely what any given student knows at any point in time relative to any given hierarchical knowledge representation (i.e., SLT rule).

Figure 5 goes further. It shows how TutorIT automatically infers what any given student does and does not know relative to any given domain. These

FIGURE 5
Shows how TutorIT makes its decisions and inferences during the tutoring process.
Deterministic AI Foundation for Modeling Human Tutors

Inferences are based on S behavior on various (partially solved) sub-problems in a domain. Not only does TutorIT pinpoint what steps a student has mastered in a problem solution but also what remains to be mastered. Patented processes also enable perfectly logical inferences about behavior on other steps and decisions required in solving problems. Neither ACT-R nor any other theory or technology I am familiar with does this.

Larger Ill-Defined Domains

Hierarchical inferencing is a perfectly general process and can be applied to any content domain. Nonetheless, some/many domains are so large that they cannot easily or reasonably be represented in terms of individual hierarchically defined SLT rules. There is a natural solution here as well.

Higher order SLT rules have been an integral part of SLT from its inceptions (e.g., see Scandura, 1970, 1971). Higher order SLT rules operate on given SLT rules and generate new SLT rules. A simple example would be a higher order SLT rule that takes two SLT rules of the form $A \rightarrow B$ and $B \rightarrow C$ and maps those rules into a rule of the form $A \rightarrow C$ (i.e., $A \rightarrow B \rightarrow C$). Such a higher order rule might be represented $(A \rightarrow B, B \rightarrow C) \rightarrow (A \rightarrow C)$. Such a higher order rule will be familiar to some under the label “chaining”. Other simple broadly applicable higher order SLT rules would correspond to generalization, analogy, etc.

It is important to emphasize that ALL SLT rules, lower and higher order alike, have a defined domain of applicability. For example, SLT rules (e.g., performing column subtraction or solving linear equations) have defined domains of application. Unlike general purpose “chaining”, for example, higher order rules also have a defined domain of application. They are NOT universally applicable. Universality is reserved for UCM (see below).

The introduction of higher order SLT rules provides a highly efficient way to account for the behavior in arbitrarily large problem domains. As summarized in Figure 6, it is not necessary for individual SLT rules to account for ALL behavior in any given domain. SLT solution rules for many, often most problems in a domain cannot be derived via any single SLT rule. Rather they must be derived (or selected from multiple candidates) via higher order SLT rules operating on other SLT rules. In this case, higher order SLT rules operate on given SLT rules to generate new SLT rules that do solve the given problems. The only difference between higher and lower order SLT rules is that the former operate on elements that happen themselves to be SLT rules.

A simple case of this is when a domain consists of a (relatively small) finite set of more or less related problems. This is often the case with word problems in
school arithmetic or algebra. Word problems in algebra are often categorized as to solution method: area, perimeter, quadratic, rate, simultaneous equations. Each requires a different solution method. Given such a domain, the first thing a S must do is to select an appropriate solution method.

The higher order rule needed in this case is referred to as selection rule. Given a problem, the first thing that must be done is to select an appropriate SLT rule. This has been done successfully in developing a TutorIT tutorial for algebra word problems. This is the only one to date, where solving a given problem requires deriving (in this case selecting) the appropriate SLT solution rule. To date, we have not done this (see Figure 6).

**Higher order SLT rules make it possible to represent essentially any kind of learning.** For example, a familiar higher order SLT might chain SLT rules (e.g., a SLT rule for converting yards into feet and another for converting feet into inches) to create a new SLT rule (i.e., for converting yards into inches, first yards into feet and then feet into inches). Another higher order SLT rule might map (generalize) rules with restricted domains into SLT rules that are more widely applicable. There is no limit on the variety of higher (or lower) order SLT rules. It all depends as needed on the domain from which they are derived.

ideas also were used to represent the contents of an entire math textbook for teachers in terms of lower and higher order SLT rules. Furthermore, experimental results demonstrated that explicit attention to higher order SLT rules made it possible to teach less (many fewer SLT rules) while students in fact learned more. Ss taught higher order rules could derive solutions to new problems they had never seen before whereas the others could not (Scandura, 1974; Ehrenpreis & Scandura, 1974).

As above, the benefits of using SLT rules and higher order SLT rules have been clear from their inceptions. Until recently, however, the processes by which lower and higher order SLT rules have been created has been largely informal. Creating a systematic method of Structural Analysis (SA) has been a major long term challenge. Scandura (1984a,b) was an important step in that direction. A full solution, however, did not come until I switched from representing SLT rules as directed graphs (think flow charts) to AST hierarchies. This switch came as a natural evolution based on our work in AI and software engineering (e.g., Scandura, 1991, 1994) in the new millennium (e.g., Scandura, 2001, 2007). Implementing these ideas in an operational system, however, remains a major challenge.

FIGURE 7
Nodes in Higher Order SLT rules are themselves SLT rules, each represented by its own hierarchy.
How IS KNOWLEDGE PUT TO USE – UNIVERSAL CONTROL MECHANISM (UCM)?

SLT’s Universal Control Mechanism (UCM) has also played a central role in SLT from its inceptions. It made it possible to both explain and predict the behavior of individual Ss with deterministic precision (Scandura, 1974). Originally, UCM was formulated roughly as follows:

“Look for a SLT solution rule that matches the input and output of that problem. If no such SLT rule is found (i.e., is available in S’s memory), look for (and use) a higher order SLT rule that generates (or recovers from memory) an SLT rule that solves the problem.”

The introduction of higher order SLT rules in experiments demonstrated their value in predicting behavior. Positive results were found in experiments run under extremely well-controlled conditions (e.g., Scandura, 1974) and in experiments covering broad more loosely defined domains involving large numbers of lower and higher order rules (e.g., Ehrenpreis & Scandura, 1974).

There were limitations however. UCM was well defined when a SLT solution rule existed:

FIGURE 8
Summarizes the requirements UCM must meet.
Given a problem UCM first seeks a SLT rule that may solve the problem: Where the range of the SLT rule matches the problem goal and the domain of the SLT rule matches the given information in the problem.

So far so good – for simple domains! However, this approach had a critical gap. What happens when no such SLT rule exists? You might guess that higher order rules are involved. But where do higher order SLT rules come from – other than the mind of the SA analyst? The way UCM was originally formulated made it impossible to completely distinguish between UCM and higher order rules.

The problem with the above is that it had a major gap. How to determine whether any given Higher Order SLT Rule can generate a needed SLT solution rule?

Our SLT solution came from an insight gained from our software engineering research. Rather than create higher order rules out of thin air, the solution lie in recursively seeking (SLT rule) matches INSIDE available higher order SLT rules themselves.

Given a problem, if no solution SLT rules match, UCM searches for (higher order) SLT rules whose range contains SLT rules that match the given problem. The ranges of these higher order SLT rules include SLT solution rules that may solve the given problem.

 Appropriately generalized, this recursive process is well-defined and readily lent itself to implementation in TutorIT software. The difference is subtle but essential and necessary to create a UCM that is independent of Higher Order SLT rules.

SOLUTIONS

Added to what we already knew based on earlier versions of SLT, these two solutions (rigorous hierarchical knowledge representation and a well-defined and implementable UCM) gave us the machinery needed to build our AuthorIT authoring and TutorIT delivery platforms (e.g., Scandura, 2016).

We can now represent to be acquired knowledge with arbitrary degrees of precision. We also can efficiently pinpoint what any student does and does not know at any point in time. This includes inferring Ss knowledge on hierarchically related but untested knowledge associated with what is to be learned. We also can pinpoint the knowledge any given student needs to progress.
• **AuthorIT**: The systematic method of SA together with human judgement (of STEM SMEs) makes it possible to systematically represent what needs to be learned for success – in terms of hierarchical SLT Rules (AST hierarchies).

• **TutorIT**: Automated diagnosis and remediation in TutorIT makes it possible to pinpoint what Ss do & do not know at each point in time relative to SLT rules as well as what needs to be learned for success.

In short, implementation of a tutoring system based on the SLT involves human judgement using AuthorIT (an implementation of Structural Analysis) combined with artificially intelligent TutorIT decision making.

As they currently stand, the AuthorIT authoring and TutorIT delivery systems collectively represent an important step forward in building adaptive learning systems. They offer a concise, well-defined combination of: a) Human judgement used in AuthorIT to identify what must be learned for success plus b) Automated decision making in TutorIT. In effect, general purpose tutoring system that works with all content. In short, human Judgment in AuthorIT is combined with AI automation in TutorIT to Model the Human Tutoring process.

AuthorIT and TutorIT technologies have proven to be highly effective and efficient technologies for creating and delivering instruction in well-defined task domains. By definition, these technologies can ensure predetermined levels of mastery. Together, they make it both possible and feasible (cost-effective) to model human tutoring processes with considerable fidelity. TutorIT quickly determines what any given S already knows, and what the S needs to progress. TutorIT also continues the process until the S either achieves mastery or gives up.

In effect, AuthorIT and TutorIT model human tutoring to an unprecedented degree. Unlike other adaptive tutoring systems, assuming student completion, TutorIT tutorials created with AuthorIT make it possible to literally guarantee predetermined levels of mastery.

How do we know this? Guaranteed mastery is literally built into TutorIT. Any given student may not complete a given tutorial, and they may forget what they have learned. Nor is there is any guarantee that what they learn will be valuable. This is determined by and depends on the author. Nonetheless, any student who completes any given TutorIT tutorial must necessarily achieve a predetermined level of mastery.

To date, published data are limited. Using an early version of EZauthor and TutorIT, Novak found that all students who completed her TutorIT tutorial in statistics demonstrated mastery of the content (Novak, 2014). John Durnin (2016) independently achieved similar results with the same tutorial. Many Ss,
however, failed to complete the tutorial because they entered without the necessary prerequisites. Making provision for and remediating missing prerequisites is now possible in the current version of EZauthor, but this is not the place to detail the process.

What we can say is that authors set mastery criteria and Ss must achieve those criteria to complete any given TutorIT tutorial. Retention over time is a separable issue that can be addressed by standard methods (e.g., spaced repetition and review).

It is instructive in this case to note that the deterministic model on which AuthorIT and TutorIT are based does not preclude attention to other instructional variables. There are many things authors have and often do to make it easier for Ss to learn and/or retain knowledge. The kinds of instruction and/or explanations given, student preferences for kinds of instruction, for example, may facilitate learning (e.g., reduce the time needed for success). Although such variables may and often do help, these variables are strictly secondary. The bottom line is whether a S has mastered the content in question. SLT, and AuthorIT and TutorIT based thereon, allow for and even encourage the inclusion of such variables. Indeed, many TutorIT tutorials use multiple aids to make it easier for student to learn. The fundamental question, however, always is what they know.

Results to Date

Several generations of student externs at Central Florida University have joined us for a week. Successive improvements enabled all recent externs to both learn how to use our authoring platforms and to create their own TutorIT tutorials, some of professional quality – all in 4 or 5 days.

To date, we have not had access to large student populations. However, many TutorIT tutorials have passed intensive internal review by subject matter experts working alone or with individual students. For example, a bright young 8 year old who had started but not completed column subtraction in school demonstrated mastery in a total of only 32 minutes.

Complex Domains Requiring Higher as well as Lower Order knowledge

Algebra word problems and Math Abilities (aka Critical reading) aside, TutorIT tutorials to date have been limited to relatively simple well-defined domains. Predicting behavior in this case is relatively straightforward. One presents whole or partially solved problems and Ss either succeed or not. In the latter case TutorIT provides the information needed at each point in time continuing until the S either masters the content or gives up.
As with word problems above, solving problems often requires more than simply applying learned techniques (e.g., SLT rules). SLT has made provision for such situations since its inceptions (Scandura, 1971).

Higher order SLT rules, SLT rules that operate on and may generate new SLT rules, play a major role in problem solving. Higher Order SLT rules have been used and tested in empirical research for many years. Until recently, however, higher order SLT rules have been represented as directed graphs (or equivalent Flowcharts). None have been represented as ASTs. And, certainly none (other than simple selection as above) have been created with AuthorIT or implemented in TutorIT.

SLT’s Universal Control Mechanism (UCM) also has played a central role in the problem solving process from inception of the SLT (Scandura, 1971). UCM controls how SLT rules, higher as well as lower order, interact in solving problems. UCM has been implemented in TutorIT but testing has been severely limited. To date it has only been used with single rules or in rule selection. A future challenge would be to develop TutorIT tutorials for complex domains requiring higher order SLT rules. The theoretical implications are both broad and deep. Nonetheless, we have barely begun to capitalize on the current capabilities of AuthorIT and TutorIT. For now, this represents just one area where the SLT remains in advance of its implementation in AuthorIT and TutorIT.

**MEMORY AND PROCESSING CAPACITY**

At this point you might ask what happened to memory and processing capacity. Both have played a foundational role in SLT from its inceptions (Scandura, 1971). One of SLT’s basic assumptions has been that each individual has a fixed processing capacity and a fixed processing speed. A series of studies confirmed a fixed processing capacity (Scandura, 1971, 1973; Voorhies & Scandura, 1977).  

---

18 It should be emphasized that testing deterministic assumptions requires specially designed experiments. Please see the above references for specifics. It is not sufficient to test under standard conditions with random subjects tested with random tasks. To the contrary, one must take great care to ensure that subjects are performing any given task in a highly prescribed manner. Controlling the way subject process information is essential. This makes it possible to accurately compute functional memory loads – for example, how many distinct elements need to be kept in STM as a task is being performed. Functioning memory load obviously can be reduced by chunking information. Precisely the same constraints would have to be followed in testing inherent processing speed, which under normal conditions will vary greatly for any given individual on various tasks. As with STM, processing speed will depend on how well mastered the task in question.
This assumption that each individual has a fixed, or at least characteristic processing speed remains to be formally tested.

No attempt has been made to measure or to use these theoretical assumptions in TutorIT decision making. These assumptions in SLT are attributed to individual Ss. Irrespective of where a particular S begins, TutorIT is designed to ensure that ALL students on completion will achieve a specified level of mastery. (This assumes, of course, that ALL students enter at minimum with assumed predetermined prerequisites.) TutorIT currently makes no direct provision for whatever differences may exist in a student’s processing capacity or processing speed. Nonetheless, AuthorIT and TutorIT do make provision for the closely related concept of automation. The more automated a S’s knowledge becomes, that larger the “chunks”. In SLT and otherwise, larger chunks lower the memory load and increase the speed with which a S can perform a given task. See the blue arrow in Figure 4A for a concrete example. In this case the task is to demonstrate the ability to regroup (borrow) across 0s in column subtraction.

The small “a15” before the node in Figure 4A (just below the blue arrow) indicates that the student must not only demonstrate mastery of the corresponding task. After demonstrating the ability to do this, the S must demonstrate a higher level of mastery. The S must not only perform the intermediate steps in his or her head but respond within the predetermined time – this is designated by “a15” (i.e., automation within 15 seconds)

FUTURE WORK

SLT offers the potential of still more. As above, there is no in principle reason why TutorIT could not be generalized to support mastery of higher order knowledge. I say this knowing that technical implementation problems will not be easy. At this point it is hard to know how feasible or how much technical work will be required. Again, I must say the same with respect to short and long term memory and processing speed.

In effect, still to be implemented futures based on the SLT include:

1. Extending Structural Analysis (SA) to support systematic representation of Higher Order SLT rules. The closest we have come to this so far is with TutorIT’s Word Problem tutorials. These tutorials require students to select appropriate from among alternative solution rules as well as to apply them.
2. Taking into account each Ss presumed processing capacity and processing speed. We can currently require automation as well as mastery by imposing
desired response times. But these currently are general requirements for all students. Theoretically, these requirements could be associated with individuals.

3. Unconstrained dialog between Student and Teacher is not supported. The closest we have come to date is allowing students to ask for and get more help.

As above, we have already implemented UCM in TutorIT. However, we are yet to make significant use of that capability and cannot until we can systematically support the construction of higher order SLT rules. Our current prototype, which involves selecting appropriate SLT rules for solving algebra word problems, is only a beginning.

In short, advanced as they are, AuthorIT or TutorIT still omit significant portions of SLT. I’ve already detailed how TutorIT makes its decisions in Scandura (2016) so I won’t repeat that here. Nor, is this the place to repeat the role and purpose of EZauthor, Customizer, Scope and Sequence or our full-featured AuthorIT authoring system. I would simply point out that neither AuthorIT nor TutorIT directly address each student’s assumed characteristic processing capacity (Voorhies & Scandura, 1977) or processing speed, much less long term memory.

These basic student characteristics are an integral part of the SLT. As such, these characteristics would impose further constraints on how tutoring might/could be adjusted to maximize learning via periodic repetition and/or review. AuthorIT and TutorIT currently address these issues partially by including provision for automation (response speed) and review to help ensure student retention.

To be sure, I don’t want to split hairs with ITS proponents who may argue that tutoring systems should allow students to do whatever they want, and simply provide help to get them back on track. Beyond the fundamental difference in the nature of the theories (stochastic in ITS versus deterministic in SLT), see Scandura (2013a) for other arguments concerning the relative merits of ITS and AuthorIT/TutorIT.

Suffice it to say that eliminating the historic distinction between declarative (structural) and procedural knowledge is a game changer with enormous potential. The ability to represent ALL knowledge hierarchically completely eliminates the need to program pedagogical decision making (in TutorIT). Without this, there is no way a small research operation could possibly have developed so many tutorials (see www.TutorITweb.com).

Another potentially important issue is support for open ended interactions between students and teachers (or any two or more individuals). Students can now request information from TutorIT, for example, but only in a limited sense.
Unconstrained dialog between students and TutorIT is theoretically possible, but implementation would require significant work. A few years back, I prepared a DARPA proposal on the subject, but the project was ahead of its time and not funded. Support would involve overlaps in the knowledge representations of students and teachers. While more operational, hence testable in nature, extension of SLT in this direction would be analogous to and add needed precision and rigor to Gordon Pask’s Conversation Theory (1976).

In conclusion, my hope is: that this theoretical note, supplemented with the last full description of SLT (Scandura, 2007) will be instructive. I also apologize for the large number of Scandura et al references. AuthorIT and TutorIT are the direct outcome of research over many years. Most of the basic research supporting SLT was conducted at Penn with the dedicated help of many of my former students. Their help is gratefully acknowledged. Essentially all of the theoretical and technical work leading to AuthorIT and TutorIT has been done after my retirement from Penn.

The good news is that AuthorIT authoring and TutorIT delivery platforms are now available at www.TutorITweb.com for both research and partnership purposes. My team and I look forward to hearing from you.

Final Remarks

In many ways the Structural Learning Theory (SLT) is not a learning theory as such. It does not purport to detail what goes on in people’s brains as they learn or solve problems. It simply makes assumptions about constraints that affect a learner’s observable behavior. In many ways SLT is a science of the artificial (cf. Simon, 1996).

What most distinguishes Structural Learning Theory (SLT) is its goal. SLT is first and foremost a deterministic theory. Its goal is to explain, predict and indirectly to control the behavior of individuals in specific situations. Deviations from prediction are attributed to noise in the system. Variables such as number or variety of examples, types of media used, time on task, even motivation – all are relevant and may affect both the rate and likelihood of learning. These factors, however, are strictly secondary. What matters most is what any given student knows at any given point in time and what he or she needs to learn for success. Other factors are strictly secondary. SLT is a learning theory that explains human

19 Insights from R&D since that time suggest further refinement of SLT – hopefully to be added at some point.

20 Several helped with more than single studies. For their help I would like to single out John H. Durnin, Wallace Wulfeck, Donald Voorhies, George Lowerre and Walter Ehrenpreis.
behavior from the perspective of an outside observer, teacher or tutor. To be sure, SLT makes assumptions about physical or mental constraints on a learner’s behavior. Observable behavior is constrained by how much (many chunks) a learner can process at any given point in time, how rapidly a learner processes information, potentially even ultra-short term constraints on memory (see “Concluding Comments and Implications” in my first introduction to SLT (Scandura, 1971, pp. 20–1).

A couple final comments are in order. Early on, I proposed automation and UCM as foundational assumptions, not just something to the effect that automation takes place via practice over time, or just another control mechanism on the order of means-ends analysis (Newell & Simon, 1972) or any of the various control mechanisms assumed in ACT-R (Anderson, 1988).

Automation fundamentally refers to the level of the operations and decisions in the SLT rule that S uses when solving a problem. In column subtraction, for example, S might answer quickly or otherwise. How does this take place? Levels in a hierarchy correspond to the knowledge an individual uses in solving a problem. One expects faster performance when S operates at a higher level in an AST-based SLT rule.

Practice has this effect. Nonetheless, S may opt to perform lower level operations explicitly. Speed differences are implicit in representing SLT rules as ASTs. All else being equal, performing at higher levels in an AST hierarchy takes less time. What are the mechanisms by which this happens? Do higher order SLT rules eliminate intermediate steps? A deeper level of explanation clearly would be desirable.

All we can say at present is that it is possible to build highly effective and efficient TutorIT tutorials without delving into issues pertaining to short term memory (STM) – even though STM has been an integral part of SLT since it’s inceptions (Scandura, 1971). Studies with Don Voorhies (Scandura, 1971, 1973; Voorhies and Scandura, 1977) showed that each individual has a fixed processing capacity irrespective of the task involved. This research required direct control over the methods Ss used while solving problems. We did not allow Ss free rein but rather ensured by careful pre-training that Ss performed given tasks in a prescribed manner, thereby making it possible to determine memory load imposed on the learner.

As a final note, we have published evidence of UCM in studies with children as young as 4. Is processing capacity something learned or built in from birth? With this in mind, I ran informal tests with my two oldest grandsons who were about the same age even before they were able to crawl. Sam was on the floor and badly wanted to get a toy. However, every time he tried to crawl, he moved backward. He grew obviously frustrated. He really wanted that toy. All of a sudden, he reversed his position, attempted to crawl, moving backward closer to the
toy. Reversing his position once again, he found the ball with reaching distance. Evidence of a higher order composition rule operating in conjunction with UCM? (In case you’re wondering, Zac’s initial higher order repertoire favored analogy.)

REFERENCES

Ehrenpreis, W. & Scandura, J. M. Algorithmic approach to curriculum construction: A field test. Journal of Educational Psychology, 1974, 66, 491–498. (Also in Problem Solving; see B-6.)
Miller, G. A. The Magic Number Seven, Plus or Minus Two. Psychological Review, 1956, 63, 81–97.


Scandura, J. M. Prior learning, presentation order, and prerequisite practice in problem solving. *Journal of Experimental Education*, 1966, 34, 12–18. (c)


Scandura, J. M., Wulfeck, W. H., II, Durnin, J. H., & Ehrenpreis, W. Diagnosis and instruction of higher-order rules for solving geometry construction problems. In 1974 *Proceedings*; see B-5. (Also in *Problem Solving*; see B-6.)


Unpublished Paper with more detailed information about memory


https://www.dropbox.com/s/arvwr8lgp95urwa/087_Algorithmic%20analysis.docx?dl=0