Dynamically Adaptive Tutoring Systems: Bottom-Up or Top-down with Historic Parallels

JOSEPH M. SCANDURA

Director of Research, Intelligent Micro Systems and Emeritus Professor
University of Pennsylvania, USA
scandura@scandura.com, web: www.TutorITweb.com

This article exposes surprisingly close historical parallels in the development of Intelligent Tutoring Systems (ITS) based on biologically inspired ACT-R theories and dynamically adaptive tutoring systems based on operationally defined cognitive constructs that serve as a foundation for the Structural Learning theory (SLT). The article begins with a comparison of geocentric Ptolemaic and heliocentric Copernican theories of the solar system. The article then traces surprisingly close parallels over the past half century in the development of ACT-R and SLT and the dynamically adaptive tutoring systems spawned by each. The former work from the bottom-up; the latter work from the top-down. Retracing history shows that the former, ACT-R theories and ITS systems based thereon, result in precise albeit complex accounts of observable behavior and ITS systems based thereon. The latter, SLT and the AuthorIT authoring and TutorIT delivery systems based thereon, result in equally precise but more cohesive and more easily constructed dynamically adaptive tutoring systems.

Keywords: Intelligent Tutoring systems, adaptive learning, tutoring systems, instructional theory, Structural learning Theory, SLT, ACT theory

1 INTRODUCTION

According to Wikipedia, the history of science until recently was seen as “a narrative celebrating the triumph of true theories over false”. Since Thomas Kuhn’s, The Structure of Scientific Revolutions (1962), scientific progress is measured more in terms of competing paradigms battling for intellectual supremacy in a broader intellectual, cultural, economic and political context.

*Corresponding author: scandura@scandura.com, www.TutorITweb.com
According to Kuhn each new paradigm re-writes the history of its science by selection and distortions of various paradigms. One of the best known examples from the past involves the ultimate triumph of heliocentric (and ultimately Kepler’s theory) over the use of epicycles to explain planetary motion in an Earth centered solar system. Given the benefit of hindsight, I show that the history of Intelligent Tutoring Systems (ITS) over the past half century has followed a similar pattern. Let me start from the beginning.

1.1 Ptolemy’s View of the Solar System

Again quoting from Wikipedia, “in Ptolemaic cosmology a small circle, the center of which moves on the circumference of a larger circle at whose center is the earth and the circumference of which describes the orbit of planets around the earth.” As observations became more precise over time, increasingly more levels of circles were introduced to match observed motion.

“As Renaissance astronomers got better at recording the exact locations of planets ..., they kept trying to plot these locations back to presumed coordinates (and offsets) on ... ‘celestial spheres’ that carried the planets. But the more precise they got, the more complicated their offsets ... (gradually becoming) celestial spheres bearing epicycles bearing epicycles bearing epicycles bearing planets.

Epicycles provided a close approximation to the data. The problem was that the situation got increasingly complex as more and more data were collected, with continuing adjustments to the theory. Increasing numbers of epicycles were necessary to compensate for data that could not reasonably be explained by adding yet-more-circles.

For almost 2000 years, Astronomers found the Earth centered Ptolemaic system to be adequate as a predictive system. It is not that a sun centered view of the solar system was unknown at the time. Epicycles and the like were not necessarily viewed as a problem. Epicycles are roughly comparable to regression formulas, such as polynomial regression. Regression formulas like contemporary psychological theories are not necessarily assumed to model every detail in reality. Rather, they are designed to match observations as closely as possible independently of cause. In complex systems we rarely know all the underlying causes. Indeed, we shall see below that the data used to support most behavioral theories falls in this category (cf. Scandura, 1971). For more specifics, please see Appendix A for convenience printed from http://csep10.phys.utk.edu/astr161/lect/retrograde/copernican.html.

1.2 Heliocentric Theory: An Alternative Sun Centered View

It is commonly believed that Copernicus’ heliocentric theory eliminated the need for epicycles. In fact, heliocentric theory reduced but did not eliminate the
need for epicycles. Epicycles were still needed to reach Renaissance accuracy levels. For a time there was no immediate or compelling need to rethink the nature of planetary motion.

It remained for Kepler’s formalisms to achieve a heliocentric, epicycle-free solar system capable of matching the detailed data collected by Tycho Brahe. Cosmology not only needed a lot of hard work but better design. Specifically, the need for epicycles disappeared with Kepler’s introduction of ellipses to the mix. Kepler’s efforts in turn would have been lost without the masterly rhetoric of Galileo Galilei to keep it afloat.

1.3 Summary

For a variety of reasons, political as well as religious, the Ptolemaic conception of an earth centered conception of the universe held sway for almost two-millennia. In this view as measurements got increasingly precise, astronomers accounted for these observations using increasingly complex refinements of existing theory, based on epicycles and offsets. In this view the planets revolved about the earth in increasingly complicated sets of circular motions within circular motions. Astronomers were able to successfully use this model to account for planetary movements in the heavens, and their varying brightness at various times of the year.

Copernicus proposed an alternative view in the 16th century (resurrecting Sun centered views from earlier times). This heliocentric, Sun centered model did not completely eliminate, but it did dramatically reduce the need for epicycles. Moreover, it created a foundation for the painstakingly precise data collection of Tycho-Brahe. Kepler’s formal accounting of the Tycho-Brahe data used ellipses instead of circles (which mathematically are a special case). The result was a much simpler accounting, and ultimately prediction.

2 SOME INTERESTING PARALLELS

At this point you are undoubtedly asking yourself “What does this have to do with Intelligent Tutoring Systems (ITS)?” The short answer is nothing directly! No historical analogy is complete. But then this short history, collapsing centuries into a few paragraphs, has interesting parallels in ITS research over the past several decades.

Consider the following. Although a sun-centric view of the solar system was not unknown at the time, early astronomers found it convenient for navigational purposes to assume that the earth is at the center of the solar system. Computer Based Instruction (CBI) based on high level analysis of what is to be learned came decades earlier than ITS. ITS researchers, however, assumed that it would be
better if instructional decisions were based on what is in learner minds. The basic assumption is that learner minds can be represented as sets of productions (condition action pairs, analogous to S-R associations) and learning mechanisms controlling their use. Assumptions had to be made about both ingredients (productions) and learning mechanisms. The latter were needed to explain how productions are used to produce behavior and/or learn new productions. The promise was more dynamic human like interaction between tutoring system and individual learners.

ITS grew up in the late 1970s and 1980s under the leadership of John Anderson at Carnegie Mellon University as an alternative to Computer Based Instruction (CBI). His biologically inspired ACT theory essentially involves an integration of his work on associative (S-R) networks and spreading activation as a Ph.D. student at Stanford under Gordon Bower with the production system brand of Artificial Intelligence championed by Allen Newell at Carnegie Mellon (derived in turn from the logician Post’s work on production systems beginning in the 1920s).

ACT theories effectively integrate what Anderson considered the best of AI with associative networks. Since that time, there have been multiple iterations of ACT theory. All build on the idea that AI theories are essentially computer programs based on production systems. Anderson’s work has evolved over the years, but to this day retains its focus on biological foundations, most recently making quantitative predictions based on patterns of activation in the brain (Anderson et al, 2008).

By way of contrast, traditional computer based instruction (CBI) had two major limitations. First, it lacked a precise theoretical foundation. Each CBI systems had to be designed and built from scratch. Second, CBI lacked the degree of interactivity offered by ITS. The long and short of it was that one size fits all CBI could not come close to the ITS promise.

In the next two sections I summarize essentials of both approaches: First, I briefly summarize the nature and impact of traditional ITS. Building an ITS begins from the bottom-up based on a method called Cognitive Task Analysis (CTA).

---

1 With early roots in the work of Sydney Pressey on teaching machines in the 1920s, Computer Based Instruction (CBI) began with programmed instruction and teaching machines in the 1960s. Two diametrically opposed views were dominant. The first influenced by B.F. Skinner centered on gradual shaping student responses through positive reinforcement. The second by Norman Crowder viewed CBI as communication, between a between a teacher and student. Branching based on learner responses played a central role. CBI as such began in the 1960s with PLATO and SOCRATES at the University of Illinois (e.g., Broudy; Stolurow), later championed by William Norris at Control Data, leading to PLATO Learning. A combination of inadequate technologies, difficulty in implementation and limited authoring capabilities (e.g., our early SURPAS authoring system in the 1970s and later AuthorWare), led to gradual disinterest in CBI as the focus of academic research.
CTA makes assumptions about ingredients in the mind and mechanisms for using those ingredients. ITS systems base their tutoring (pedagogical) decisions on those assumptions. Second, I summarize recent advances and technical developments based on the Structural Learning Theory (SLT). In contrast to ITS, this approach works from the top-down using a method called Structural Analysis (SA) (Scandura, 2007). SA is used to systematically identify to be learned lower and higher order knowledge. Knowledge in SA is initially identified in general terms and then systematically refined. As we shall, this top-down approach - like the Copernican view of the solar system - is not new. What is new is the ability to make those high level representations as precise as one needs or desires, and to do so more systematically and simply than by using CTA.

3 ACT-R AND ITS IMPACT ON ITS

By most accounts the history of Intelligent Tutoring Systems (ITS) began with Brown & Burton’s (1978) bug oriented tutoring systems in the mid-late 1970s focused on correcting common student errors. Shortly thereafter, ITS adopted a more comprehensive and rigorous theoretical foundation in Anderson’s ACT theory (see Appendix C).

To date, the only ITS to have made significant commercial inroads are Carnegie Learning’s Intelligent Tutoring Systems (ITS), otherwise called “Cognitive Tutors” (e.g., Ritter, 2005). These ITS are based on the biologically inspired theories of Anderson (1993) along with extensions by his colleagues (Anderson, Koedinger et al, 1995; Koedinger, 2007). Development methods and the resulting tutorials have gradually evolved over a period of many years largely as a result of federal subsidies in the 10s of millions of dollars.

Various iterations of ACT theory, most recently ACT-R have guided several decades of empirical research on ITS. ACT-R is uniformly recognized as having guided the development of the math ITS systems currently marketed to schools by Carnegie Learning. As both a professor and one of Carnegie Learning’s founders, Anderson devoted his career to building a complete model of human cognition and performance. While used to explain observable behavior, his theory is grounded in biology of brain imaging with the goal of understanding computational underpinnings of the human mind.

---

2 The historical record shows that essentially the same diagnostic model was used years earlier in studies by Scandura (1971) and Durnin & Scandura (1973 & 1974). Going into that here would detract from the current analogy, and it is mentioned solely to clarify the historical record and to emphasize the complexities inherent in scientific development.
According to Anderson et al (2004, p. 1036) his ACT-R (Adaptive Theory of Control-Rational) “has evolved into a theory that consists of multiple modules but also explains how these modules are integrated to produce coherent cognition. The perceptual-motor modules, the goal module, and the declarative memory module are presented as examples of specialized systems in ACT-R. These modules are associated with distinct cortical regions. These modules place chunks in buffers where they can be detected by a production system that responds to patterns of information in the buffers. At any point in time, a single production rule is selected to respond to the current pattern. Subsymbolic processes serve to guide the selection of rules to fire as well as the internal operations of some modules. Much of learning involves tuning of these subsymbolic processes.”

Does this have a familiar ring? ACT’s potential as a means of explicating complex human behavior from a biological (cf. geocentric) perspective is not at issue. The question is whether ACT-R is the best or even a good way to explain, predict and (more important from an instructional perspective) promote student learning. Admirable as are ACT’s lofty goals, understanding complex human behavior in terms of brain science is necessarily very complex. Adding the teaching and learning process only makes the situation more complex. Explaining complex human behavior in terms of brain activity is not unlike understanding chemistry in terms of its underlying physics, or biology in terms of its chemistry.

In recent expositions of ACT-R, Anderson et al (2004) presents a number of simple and complex empirical examples to illustrate how the modules they postulate function singly and in concert. Application to ITS is among the most complex. Indeed, building ITS for school learning has presented so many complications that a good deal of the Carnegie Learning curriculum involves the use of standard workbooks, in some cases inches high. The question here is whether ACT-R offers a practical, necessary or even desirable foundation for building dynamically adaptive tutoring systems.

To avoid any chance of misinterpreting, I quote directly from the Carnegie Learning website. As explained by Steve Ritter, a close associate of Anderson and co-founder with Anderson and Koedinger of Carnegie Learning:

“Anderson’s model, called ACT-R (Adaptive Control of Thought – Rational: Anderson, 1990, 1993; Anderson & Lebiere, 1998; Anderson, Bothell, Byrne, Douglass, Lebiere & Qin, 2004), is now recognized as the most comprehensive description of how the mind works, and has been the basis of thousands of

---

3 Recent extensions of the ACT theory add further complications. See excerpts from Ball (2011) in Appendix C.
publications (see http://act-r.psy.cmu.edu/publications/index.php). The ACT-R theory is regularly used to model and predict important characteristics of human behavior, including error patterns and response times in studies of a variety of cognitive tasks. The Cognitive Tutors represent an effort to apply this knowledge of how people learn to mathematics (Ritter et al., 2007).”

Ritter summarizes key tenets important to education (Anderson, 2002; Koedinger, Corbett & Perfetti, 2010). To again quote:

• There are two basic types of knowledge: procedural and declarative. Declarative knowledge includes facts, images, and sounds. Procedural knowledge is an understanding of how to do things. All tasks involve a combination of the two types of knowledge.
• As students learn, they generally start out with declarative knowledge, which becomes proceduralized through practice. Procedural knowledge tends to be more fluent and automatic. Declarative knowledge tends to be more flexible and usable in a wider range of contexts.
• The knowledge required to accomplish complex tasks can be described as the set of declarative and procedural knowledge components relevant to the task.
• Knowledge becomes strengthened with use. Strong knowledge can be remembered and called to attention rapidly and with some certainty. Weak knowledge may be slow, effortful, or impossible to retrieve. Different knowledge components may represent different strategies or methods for accomplishing a task (including incorrect strategies or methods). The relative strength of these components helps determine which strategy is used.
• Learning involves the development and strengthening of correct, efficient, and appropriate knowledge components.
• There are strong limits on students’ ability to reason. These limits are referred to as “working memory capacity.” As knowledge becomes more proceduralized, it takes up less working memory.

To reduce the chance of misunderstanding, Ritter adds:

“It is important to understand that the terminology used here differs somewhat from the same terms as they are often used in an educational context. For example, a “procedure” in ACT-R is simply a component of knowledge that can produce other knowledge components and/or lead to external behavior.”

---

4 Allowing for the generation of new productions (knowledge components) is reminiscent and historically came some years after the introduction of higher order rules in the Structural Learning Theory (Roughead & Scandura, 1968; Scandura, 1971, 1973, 1977).
On the other hand, mathematics educators might refer to the “procedure” of solving a linear equation. An ACT-R model of that task would consist of many procedures and facts. Even a simple task like adding integers may consist of many knowledge components, including ones associated with recalling arithmetic facts, executing counting actions, etc. (c.f. Lebiere, 1999).

Analogous to casting off heliocentric theory in favor of epicycles and offsets? Ritter continues:

ACT-R views learning as a process of encoding, strengthening and proceduralizing knowledge. This process happens gradually. New knowledge will be forgotten (or remain weak enough to stay unused) if it is not practiced, and elements of knowledge compete to be used, based on their strength (Siegler & Shipley, 1995). Because the ability to perform a task relies on the individual knowledge components required for that task, education is most efficient when it focuses students most directly on the individual knowledge components that are weakest in their brains.

The interaction between declarative and procedural knowledge leads to an emphasis on active engagement with the conceptual underpinnings of procedures, so that students appropriately generalize this knowledge.”

Although still active ITS development largely went out of fashion because ITS’s are difficult and time consuming to build. They have had limited impact on mainstream education (Lajoie & Derry, 1993). Disappointing results led many to suggest … “the appropriate role for a computer is not that of a teacher/expert, but rather, that of a mind-extension ‘cognitive tool” (Derry & Lajoie, 1993, p. 5). Cognitive tools rely on the learner to provide the intelligence, not the computer.

Although not often part of the discourse, another characteristic of the Carnegie Math Program is not often discussed. While major portions of the Carnegie Curriculum are functioning ITS, volumes of accompanying workbook material are needed to complement same. These materials mainly address declarative knowledge, which is not easily accommodated in ITS within the ACT-R framework.

Despite these limitations, the traditional ITS community has only rarely (e.g., Ohlsson & Mitrovic, 2007; Paquette, 2007; Scandura, 2007; Scandura et al, 2009) participated in serious scientific dialog including alternative approaches. None of this, of course, diminishes the potential importance of tutoring systems that can dynamically adapt to student needs as do good human tutors – if only they could be developed in a more cost-effective manner. Focus in ITS on what is going on
in student brains (e.g., Anderson, 1993, Koedinger, 2006, 2009) requires making fundamental assumptions about the cognitive elements (“productions”) associated with the knowledge to be acquired - and on the learning mechanisms governing their use. This is a very complex task. Indeed, the complexities involved in integrating and drawing inferences from individual knowledge components is reminiscent of the complications inherent in adding indeterminate numbers of epicycles.

Dick Clark (2012), for example, has found that Cognitive Task Analysis (CTA) traditionally used in developing ITS can miss over 70% of what is important. A good deal of what is missed pertains to how individual bits of knowledge are used in solving problems. The task is not unlike trying to reconstruct a hay stack from the individual pieces of straw.

Having to make instructional decisions as to what hints and other instruction to give significantly adds to the complexity. Variations such as using constraints instead of productions (Mitrovic & Ohlsson, 2007) help in some ways but they have been challenged both by ITS purists (e.g., Koedinger, 2009; Weitz et al, 2012) and by other promising approaches (cf. Scandura, 2007; Scandura, Koedinger, Ohlsson & Paquette, 2009). Recent attempts in ITS to work at higher levels of abstraction (e.g., Ritter et al, 2006, Paik et al, 2010) are in the right direction but in my opinion do not go nearly far enough.

A major complication is that instruction is inextricably related to the semantics (meaning) of the content. Each tutoring system requires its own unique pedagogical logic, which adds significantly to development costs. As a consequence, the effectiveness of ITS can only be determined through time consuming and costly experiments.

General purpose systems enabling the efficient development of dynamically adaptive tutoring systems have until recently seemed out of reach. ITS that naturally scale to indefinitely large content domains have been unthinkable, much less ITS that can in principal guarantee learning. In this context it is interesting to look back on a quote from the ITS community over a quarter century back (Anderson, Boyle & Reiser, 1985):

“Cognitive psychology, artificial intelligence, and computer technology have advanced to the point where it is feasible to build computer systems that are as effective as intelligent human tutors. Computer tutors based on a set of pedagogical principles derived from the ACT theory of cognition have been developed for teaching students to do proofs in geometry and to write computer programs in the language LISP.”
NOTE: Initial applications of SA in SLT focused on higher as well as lower order knowledge (cf. Scandura, 1971, 1973, 1977 & 2007). Among the more complex domains analyzed were straight edge and ruler constructions in geometry (Scandura, Durnin & Wulfeck, 1974), constructing algebraic proofs (Scandura & Durnin, 1977) and Piagetian conservation. (Scandura & Scandura, 1980).

4 STRUCTURAL LEARNING THEORY (SLT) AND ITS IMPACT ON DYNAMICALLY ADAPTIVE (AKA “INTELLIGENT”) AND CONFIGURABLE TUTORING

In a recent talk at a TICL symposium at AERA in Vancouver (2012), I made some bold claims to the effect that recent advances in AuthorIT and TutorIT (based on the Structural Learning Theory, SLT, cf. Scandura, 1971, 1973, 2007, 2011) solve three fundamental problems in traditional Intelligent Tutoring Systems (ITS). While ITS potentially have an important role to play in American education, they have had limited impact to date due to:

1. Their high cost of development,
2. The need for extensive empirical testing during development and
3. Difficulty in scaling to increasingly complex content.

Instructional decision making in traditional ITS systems based on biologically inspired theories is a very complex task.

AuthorIT & TutorIT offer a highly efficient system for developing and delivering dynamically adaptive, otherwise known as Intelligent Tutoring Systems. TutorIT is a general purpose tutor that takes a representation of what is to be learned as input and makes all diagnostic and remedial decisions automatically based entirely on the structure of what is to be learned - completely independent of content semantics. TutorIT also can easily be configured to define a wide range of delivery methods. TutorIT tutorials significantly reduce the need for empirical testing during development. They are by definition based on what subject matter experts (SMEs) believe must be learned for success. They also have the potential to guarantee mastery of the skills taught. Finally, recent technical advances make it possible to support arbitrarily complex content.

The Structural Learning Theory (SLT) rests on a fundamentally different set of assumptions than ACT-R. ACT-R and SLT both start with a content domain. Rather than making assumptions about what might be student minds, however, the focus in SLT is on what must be learned for success.
To-be-learned knowledge in SLT can be represented with equal precision, but not in terms of assumed productions along with postulated learning mechanisms. Rather, all knowledge in SLT, including higher as well as lower order knowledge, is represented in terms of SLT rules. From a theoretical perspective, **SLT rules are cognitive constructs that represent both structural (declarative) and procedural knowledge - simultaneously at all levels of abstraction** (Scandura, 2007, 2011). SLT rules are operationally defined cognitive constructs derived from to be acquired observable behavior. No commitment is made as to any particular neuronal foundation.

SLT’s method of Structural (domain) Analysis (SA) plays a central role in the process. SA is analogous to but fundamentally different than Cognitive Task Analysis (CTA) in ITS research. SA is a systematic method for identifying sets of SLT rules that collectively are sufficient for solving problems in a given domain. Unlike CTA, SA works from the top down.

The 70% gap Clark found in traditional CTA is filled in SA by two kinds of higher order knowledge. SA is systematically used to identify: a) Higher order SLT rules that operate on other SLT rules and b) multiple levels of abstraction in any given SLT rule hierarchy. Higher order SLT rules have played an essential part of SLT since it’s inceptions in 1971 (Roughead & Scandura, 1968; Scandura, 1971b, 1971c, 1974, 1977; Scandura, Durnin & Wulfeck, 1974). They play a central role in solving novel problems. In this case higher order SLT rules operate on other SLT rules and generate new ones as needed.

Higher order SLT rules are systematically derived during Structural Analysis (SA) from lower order SLT rules associated with well-defined prototypic domains within larger potentially ill-defined domains (e.g., Scandura et al, 1974). Prototypic domains effectively serve as starting points in analyzing complex domains.

The remainder of the gap noted by Clark has to do with higher levels of representation within any given SLT rule hierarchy. These higher levels correspond to increasingly more automated operations, wherein increasingly simpler procedures operate on increasingly complex data structures. Automation in this case results from practice. Automation is not a new phenomenon. What has been missing is not the process of automation. **What is new is how to represent the knowledge associated with increasing degrees of both precision and automation.**

What has been missing in SLT rules until recently is precision equivalent to productions. Productions correspond to the lowest level operations in SLT rules. **The basic approach in SA is top-down rather than bottom-up.** SA begins by simply assigning names to types of problems and to solution methods (SLT rules) for solving each problem type. I have demonstrated elsewhere how both problems and SLT solution rules can be refined systematically and indefinitely using a small number of refinement types (e.g., Scandura, 2007).
The example below deals with simple column subtraction, but the principles involved are completely general. Consider Figure 1 extracted from Scandura (2007, p. 179). This figure illustrates a very close relationship between data refinement and process refinement. At the highest level of abstraction we have a single operation \((\text{subtract})\) operating on a subtraction problem \((\text{Prob})\) taken as a whole. In this case one can view \(\text{Prob}\) as corresponding to what is in a learner’s perceptual buffer once the domain has been fully mastered. (This corresponds to “compilation” in the bottom-up approach taken in ACT-R.). In effect, “\(\text{subtract}\)” in SLT would represent full knowledge of column subtraction. It operates on subtraction problems and generates differences - effectively as perceptual wholes. “\(\text{subtract}\)” is simple fully automated operation operating on a complex “\(\text{Prob}\)” data structure. “\(\text{subtract}\)” in turn can be refined into a loop, with a condition (whose value at each point in time determines whether to repeat or exit), and an operation to be performed – in turn on each “\(\text{Column}\)”. The operation for subtracting a column, “\(\text{subt-c}\)”, is refined among other operations into an “IF .THEN” operation and “\(\text{subt-fact}\)”. The IF..THEN operation corresponds to refining subtraction columns into those where the top numeral is greater than or equal to the bottom numeral, and when it is not.

Loop and selection refinements as well as sequence (and parallel) refinements are strictly hierarchical in nature and well understood. Accordingly, they offer a natural basis for pedagogical decision making. Nodes higher in a hierarchy are by definition more encompassing (and hence intrinsically harder) than lower level nodes. This offers a natural as yet systematically unutilized foundation for pedagogical decision making in dynamically adaptive tutoring systems.
The problem is that many operations and decisions do not lend themselves to hierarchical refinement. As in Figure 1, they correspond either to conditions or parameters of operations. “*subt-fact*”, for example, operates on and generate numerals. As anyone who has worked with children knows, children do not come into the world knowing how to read and write numerals like the digit “5”.

Similarly, there is no one way to teach relationships. Unlike hierarchical refinements, there is no universal guiding pedagogical decision making. It is, however, a mathematical fact that every relationship can be represented by an equivalent operation (a mathematical function). Deciding whether one number is greater than or equal to another, for example, can be determined by treating the relationship as an operation. This operation can be refined as any other. For example, the greater than or equal operation can be refined by representing each number by a set of objects, pairing the objects one-to-one and determining which set has more objects. In effect, one can always represent a relationship as an operation. Such operations in turn can systematically be refined hierarchically just as any other operation.

In effect, one can always preserve the hierarchical structure of to-be-learned knowledge by refining parameters and relationships into operations. As above, the parameter “*Difference*” is refined into an operation for constructing numerals from straight and curved line segments. Decisions can effectively be treated in the same way. Good teachers often do the same thing intuitively. *What we have done is made this process explicit, guaranteeing that all knowledge can be represented hierarchically.*

As above, higher order SLT rules have played a central role in SLT since its inceptions in the early 1970s. Higher Order SLT rules are simply SLT rules that operate on other SLT rules and/or create new ones. **Higher Order SLT rules play a role directly comparable to learning mechanisms in traditional ITS** (e.g., ACT) theories. The main difference is that rather than assumed to be built in, higher order SLT rules can systematically be derived from SLT rules initially derived from a given domain. Hence, higher order SLT rules in SLT are domain specific. Higher order SLT rules can be used to derive new SLT rules for solving problems that fall outside the domain of any initial set of SLT rules.

Over time, various learning mechanisms have been proposed in production based theories. Learning mechanisms are assumed to be built in. They range from mean-ends analysis in Newell & Simons’ original work (1972) to chaining, to generalization (AKA Case Based Reasoning) and beyond. Each such learning mechanism corresponds to one basic type of higher order SLT rule. Higher order SLT rules are not built in, but rather are derived from given domains. Given a problem domain, higher order SLT rules are systematically identified via SA. Rather than being hard coded, as in standard ITS, higher order SLT rules are derived from the domain being analyzed.
The question then becomes are there any control mechanisms in SLT? Surely, there must be some means of controlling the use of available SLT rules. Key to the puzzle is SLT’s Universal Control Mechanism (UCM) (Scandura, 2007, 2011). UCM is totally independent of content and has been shown to be sufficient in all domains. This universally available UCM makes it possible to add, remove or modify SLT rules as desired to better reflect needs – without any change to the basic system. Compare this with even modest changes to learning mechanisms in traditional ITS, changes that often fundamentally affect the behavior of any tutoring systems based thereon. Empirical research demonstrates that UCM is and can reasonably be assumed to be universally available across all individuals (e.g., Scandura, 1971, 1973, 1974, 1977).

SLT also postulates and research demonstrates that each individual has a fixed capacity for processing information (cf. Scandura, 1971; 1973, 1977, 2007). SLT similarly assumes a fixed processing speed for each individual although this hypothesis remains largely hypothetical. While detailed exposition is beyond the scope of this article, a substantial body of research and theoretical analysis has established SLT as a generally applicable, formally rigorous, highly cohesive and operational theory (e.g., Scandura, 2001, 2007, 2009).

5 AUTHORIT AND TUTORIT BUILD DIRECTLY ON THESE THEORETICAL ADVANCES

AuthorIT’s AutoBuilder component systematizes Structural Analysis, making it possible to represent knowledge with whatever degree of precision may be desired (Scandura, 2007, 2011). Authors can in principle make contact with knowledge available to even the weakest members of a student population – and consequently make it possible to guarantee learning. The fact that all knowledge can now be represented hierarchically makes it possible for all diagnostic and remedial decisions to be made independently of content semantics.

The importance of top-down hierarchical analysis in developing dynamically adaptive tutoring systems is hard to overemphasize. AuthorIT’s support for strictly hierarchical knowledge representation dramatically reduces development costs and times. Similarly, TutorIT’s ability to make all pedagogical decisions independently of content semantics also reduces the need for empirical testing. Not inconsequentially, strictly hierarchical knowledge representation holds potential for guaranteed mastery of basic skills.

How is this accomplished? Instead of representing knowledge in terms of discrete productions (and learning mechanisms), all knowledge in SLT is represented in terms
of hierarchical Abstract Syntax Tree (AST) based SLT rules. SLT rules represent all knowledge, procedural and structural (declarative) alike, simultaneously at all levels of expertise. Given a content domain, AuthorIT’s AutoBuilder component is used to create arbitrarily precise AST-based representations of what must be learned for success.

Moreover, Scandura (2003, 2007) shows that this can be accomplished via a small finite number of data and corresponding procedural refinements. Suffice it to say here that component, category and dynamic data refinements correspond to parallel, selection (e.g., IF..THEN, LOOP) and interaction (e.g., callback) procedural refinements. Such refinements can in principle be continued indefinitely (Scandura, 2007, 2011). This basic fact ensures that terminals correspond to minimal requirements for any targeted student population.

SLT rules and higher order SLT rules represent what must be learned for success in the domain in question. Both data and processes in SLT rules are represented as Abstract Syntax Trees (ASTs), in which procedural ASTs operate on problem data, which also is represented as ASTs.

In short, SLT rules represent both the structural and procedural knowledge, various portions of which will be available to individual students at each point in time. Expert knowledge is primarily structural in nature. The operations are relatively simple and operate on relatively complex data structures. Conversely, neophyte knowledge is relatively more procedural in nature, wherein operations and decisions operate on simpler data structures.

The analogy here is straightforward, and has a direct counterpart in software engineering: Neophyte knowledge in SLT involves relatively complex procedures operating on relatively simple data structures. Expert knowledge involves relatively simpler procedures operating on relatively complex data structures. How much is structural and how much procedural in SLT is simply a matter of degree at each point in time as learning progresses.

Higher levels of mastery are relatively more structural in nature, and conversely, lower levels of mastery are more procedural. In effect, procedural knowledge is gradually converted into structural (declarative) as learning progresses. Acquiring expertise in SLT is viewed as converting procedural to structural knowledge (conscious knowledge to automatic encoding and decoding). Note: According to Ritter (quote above), this is precisely the opposite of what is assumed in ACT theories.

TutorIT is a general purpose tutor that takes hierarchical AST-based (SLT rule) representations of what is to be learned as input and makes all diagnostic and remedial decisions automatically. Higher levels in an AST-based hierarchy represent higher levels of mastery. Accordingly, if a student demonstrates mastery at a higher level of abstraction, one can assume mastery of all lower level knowledge on which it depends. Conversely, failure implies failure on all higher levels on which it is based.
Accordingly, pedagogical decision making becomes highly efficient, and is based entirely on the structure of what is to be learned. Decision making is completely independent of content semantics. As an added bonus TutorIT can easily be configured to define a wide range of delivery methods (e.g., see Scandura, 2007, 2011), ranging from serving as a simple performance aid to highly adaptive tutoring with efficient diagnostic and/or discovery learning in between.

Authors have the option of representing to-be-learned knowledge with arbitrary degrees of precision. This ensures they can make contact with prerequisites available to all students in a given population. Each TutorIT tutorial is based on an arbitrarily precise representation of what needs to be learned for success. Unique patented technologies make it easy to pinpoint what any given student knows at each point in time. It also pinpoints the information individual students need to progress, when they need it.

The precision with which to-be-acquired knowledge can be represented, together with available delivery options, makes it possible to guarantee learning. Any student who enters a TutorIT tutorial with the predefined prerequisites and who completes the TutorIT tutorial will by definition have mastered the skill to the specified level of assurance! The empirical question is not whether a student will succeed, but rather how long it will take and whether the student will be motivated to complete any given tutorial.

Recent AuthorIT extensions also support the representation of higher order knowledge. The strictly modular nature of SLT rules and higher order rules makes it possible for TutorIT to support arbitrarily complex content. In this case initially constructed SLT rules serve as data for further analysis. Previously identified SLT rules serve as data used in SA to identify higher order SLT rules sufficient for deriving (i.e., generating) entire sets SLT rules of rules of the same genre (e.g., Scandura, 2005, 2007, 2011). Higher order SLT rules are used, for example, to decide which SLT rule to use when more than one higher order rule may apply (e.g., as required in solving various kinds of word problems). They also are used to derive new SLT rules from known SLT rules as needed to solve problems. Chaining two or more SLT rules offers a simple, but hardly exclusive prototype higher order rule (see Scandura, 2007, for further discussion).

Ongoing improvements are still being made to both AuthorIT and TutorIT. Among other things our original TutorIT and AuthorIT desktop applications are

---

5 To be sure a variety of shell-based authoring systems have been developed (e.g., XAIDA, Merrill’s ID2). Traditionally, these offer different instructional strategies for different categories of learning (e.g., XAIDA was directly motivated by Gagne’s, 1985, taxonomy of learning categories). Unlike TutorIT, however, these shells are severely limited in both scope and interactivity.
now available on the web at www.TutorITweb.com and www.AuthorIT.net. Development times (and costs) have been a small fraction of that in ITS development with continuing reductions based on continuing refinements and extensions. AuthorIT has successfully been used in recent years to build highly adaptive and configurable tutorials for a growing number and variety of TutorIT tutorials. The following are now ready for field testing: Basic Facts, Whole Number Algorithms, Fractions, Decimals, Signed Numbers (Integers), Complex Expressions (with parentheses), Basic Math Processes, Logical Thinking (a comprehensive package based directly on a published workbook series on Critical Reading), Linear Equations, Simultaneous Linear Equations, Quadratic Equations. A comprehensive Tutorial on solving algebra word problems is currently under development.

Another very recent advance is even more exciting. AuthorIT and TutorIT can now be used to develop and deliver highly adaptive and configurable tutorials for essentially any content, whether in mathematics, other STEM domains, test preparation, reading, other school subjects or business training. In the process development times and costs have been further reduced - to as little as a day when existing content is available.

To summarize, the AuthorIT authoring system makes it possible to cost effectively develop dynamically adaptive tutoring systems - in a small fraction of traditional times. AuthorIT is used to create a precise representation of what must be learned to master a given body of content. TutorIT is a general purpose delivery system. It takes the output of AuthorIT as input, and makes all diagnostic and instructional decisions automatically during the course of instruction as might a good human tutor.


Given overlapping concepts and terminology, it has taken nearly half a century to fully understand the implications of using SLT vs. ACT-R as a foundation for intelligent tutoring. In retrospect, the differences are stark. Representing knowledge from the top-down as in SLT versus the bottom as in ACT-R makes it possible for the first time to completely separate knowledge representation from pedagogical decision-making. Knowledge compilation in ACT-R involves converting declarative to procedural knowledge. Automation in SLT involves converting procedural to structural (declarative) knowledge. The half century it took to clarify these
distinctions is a mere blip in the calendar. It took a millennium for heliocentric century to successfully compete with an entrenched Polemic view of the solar system.

6 COMPARING COGNITIVE TASK ANALYSIS (CTA) AND STRUCTURAL (DOMAIN) ANALYSIS (SA)

As we have seen, knowledge representation plays an essential, undoubtedly the single most important role in both traditional ITS based on ACT-R and intelligent, dynamically adaptive TutorIT tutoring systems based on the Structural Learning Theory (SLT). Cognitive Task Analysis (CTA) and Structural (domain) Analysis (SA) are names given to the methods used to construct to be learned knowledge.

6.1 Cognitive Task Analysis (CTA)

Starting from biological perspective, the main goal of CTA is to identify productions needed in building an ITS. One must identify all components that are or might be used. In ACT-R, these components are to-be-learned productions. These productions traditionally have included production rules leading to common errors. The difficulty here is that it is very difficult to identify all of the productions that might be needed in any non-trivial domain. The above estimates by Clark (2012), for example, suggest that 70% of the knowledge necessary for success normally goes unspecified.

In addition, it is necessary in ACT-R to identify the learning mechanisms governing the way productions are used. Various mechanisms have been proposed over the years. Initially, Newell and Simon proposed means-ends analysis as a single uniform learning mechanism. This later morphed into chaining, followed by a variety other mechanisms proposed by various investigators, analogical or case based reasoning (CBR) being a favorite of those committed to constructivist theory. ACT-R also proposes a variety of buffers along with production matching and selection mechanisms (i.e., control mechanisms). In addition, ACT-R devotes considerable attention to constraints on working memory (e.g., Anderson, 2007; Ball, 2011). Such constraints play at best only an incidental role in current ITSs (Ritter). Hence, further discussion here is limited to listing contrasts. Despite this complexity, contemporary ITS based on ACT-R have very little if anything to say about declarative, or what ACT-R views as factual knowledge.

6.2 Structural Analysis (SA)

Structural Analysis (in the past sometimes also confusingly referred to as “cognitive task analysis” to distinguish it from original behavioral forms of task analysis introduced by Robert Mager in 1958 and popularized by Robert Gagne
in education in the 1960s) has played a central role in SLT from its inceptions in 1971 (Scandura, 1971b, 1973, 1977). SLT rules initially were represented as Flowcharts (or their formal equivalent, directed graphs). Directed graphs may be viewed as a sequence of actions (operations) to be performed and decisions that must be made during the course of solving problems. Problems in well-defined domains can typically be solved by single SLT rules. As above, solving problems in more complex, often ill-defined domains typically requires higher as well as lower order SLT rules. Higher order SLT rules are typically used to derive new SLT rules as needed to solve novel problems in ill-defined domains.

Directed graphs have a lot of advantages. They make it possible to efficiently pinpoint areas of student strengths and weaknesses, and to identify gaps that need to be filled. Unfortunately, however, no one level of analysis works well for all students. What we did in those days was to represent SLT rules/directed graphs/flow charts in terms of what we called atomic rules. Atomic rules were viewed as so simple that the operations and decisions involved could only learned in all-or-none fashion (by students in the target population). So-called declarative knowledge was simply a degenerate form of an SLT rule, as were associations (one input-one output), concepts (many-to-two, exemplars and non-exemplars) and procedural knowledge (many-many mappings).

To make a long story short(er), our work in software engineering led to the recognition that a variation on Abstract Syntax Trees (ASTs) used in compiler theory offers an ideal solution to several issues: AST-based SLT rules are based on a unique integration of hierarchical data analysis (used in designing data bases) and structured analysis used in procedural software design. So defined, SLT rules make it possible to represent all levels of (knowledge) abstraction simultaneously. This solves several fundamental problems, not the least of which is “what is the proper level of (knowledge) representation?” AST-based SLT rules also eliminate the artificial dichotomy between procedural and declarative knowledge. **ALL knowledge is simultaneously both structural and procedural in nature. SLT rules represent both declarative (I prefer the term structural) and procedural knowledge simultaneously at all levels of abstraction.** Neophyte knowledge involves more complex procedures operating on simpler data structures. Expert knowledge involves simpler procedures operating on relatively more complex structures. This representation also has direct application to software engineering and has been used in defining the High Level Design (HLD) language used in AuthorIT and TutorIT.

Structural Analysis (SA) may be continued indefinitely. This effectively ensures that all operations and decisions can always be refined sufficiently to make contact with entry capabilities of even the weakest members of the target
student population. The bottom line is that AST-based SLT rules make it possible to completely eliminate the need for programming pedagogical decision making. **Making pedagogical decisions automatically is the single most important benefit to be drawn from our research.** This has made it possible to develop a general purpose tutoring system that makes all pedagogical decisions independently of content semantics. This effectively eliminates one of the most difficult time consuming tasks in ITS development. Indeed, recently patented methods building on SA eliminate the need for programming. They make SA, and hence TutorIT authoring accessible to otherwise computer literate subject matter experts with little or no programming experience.

The highly systematic, top-down approach inherent in Structural Analysis (SA) is precisely the opposite of Cognitive Task Analysis (CTA) in ACT theories. SA starts from the top down, no matter how complex the domain. There is no need to make assumptions about whatever may reside in learner brains. SLT rules, and higher order rules in SLT are simply cognitive constructs, albeit constructs that may operate on other SLT rules as well as perceived data in the external world (cf. Scandura, 1971, 1973, 2001, 2007).

Incompleteness as such is never a problem in SA. The question here is not coverage as such. TutorIT works with the results of any SA. It is simply a matter of how detailed any given SA can be made given available time and resource constraints. The more precise and complete any given SA, the closer one can come to guaranteeing learning. Moreover, unlike CTA, SA can be incrementally refined, without revisiting the entire structure of the tutoring system itself.

All this is not to say that others have not worked to reduce the complexity of traditional ITS. Ohlsson & Mitrovic (2009), for example, have used Constraint Based Modeling (CBM) to bypass productions in favor of constraints. Rather than trying to adjust instruction based on assumed actions (productions) in human brains, CBM reduces complexity by focusing on assumptions about (declarative) constraints used by learners to assess situations and their satisfaction (see Ohlsson & Mitrovic, 2009; Scandura, Koedinger, Ohlsson, & Paquette, 2009).

Ohlsson (in Scandura et al (2009, p. 134-5), however, confuses SA with the way subject matter is organized. Accordingly, he explicitly rejects the notion that one can base instruction on what subject matter experts (SMEs) believe is important. SMEs are typically concerned with how to organize subject matter from an academic perspective. In contrast, SA begins with a problem domain consisting of problems the SME analyst believes students must learn to solve. Whether the problem domain involves problems associated with the “New Math of the 1960s or solving real world problems is strictly the author’s choice. SA simply provides
a highly systematic method for identifying what must be learned for success in the given domain. Hierarchical representation resulting from SA is very different from subject matter structure in the traditional sense.

6.3 Other Contrasts

In ACT theories, automatization (or what ACT theorists call proceduralized learning) corresponds to an equally well known process in software development. Rather than representing knowledge at higher levels of abstraction, expertise in ACT is viewed in terms of compilation. The difference with SLT is stark, equally as dramatic as the contrast between geocentric versus solar centric views of planetary motion.

Just as early heliocentric thought evolved in parallel with geocentric theory, SLT evolved in parallel with ACT’s predecessors (e.g., associative networks studied by Anderson’s mentor Gordon Bower, & production systems introduced in the 1930s by the logician Post & used in Alan Newell’s theorizing). In both cases, human knowledge plays a central role. SLT, on the other hand, evolved from earlier research aimed at understanding the new math (cf. Scandura, 1964a,b; 1971). I personally was more influenced at the time by the work of Chomsky (1968), Gagne (1965) & later Pask, 1975). It’s interesting to note that behavioral task analysis (deriving from early work by Robert Miller in the late 1950s) focused entirely on observable behavior. Starting with a given task, task analysis involved systematically identifying to-be-learned prerequisite tasks from the top down in the form of a hierarchy.

I stress the word “behavioral” task analysis, not because that term was used in those early days, but rather to distinguish it from “cognitive” task analysis or Structural Analysis (SA) which focuses on to-be-learned cognitive processes. Historically, SA came before Cognitive Task Analysis (CTA). This historical fact is an oddity resulting from two camps working largely in isolation. What is now obvious but not so at the time is that SA works from the top-down whereas CTA works from the bottom up.

Recognizing that there was more to task analysis than observable behavior, I made a sharp distinction early on (e.g., Scandura, 1971) - between observable behavior and underlying cognitive processes that make that behavior possible. This distinction is important. Ignoring efficiencies and the like if there is one way to accomplish any given task there must be an indefinitely large number of other ways of producing the same behavior.

Another way knowledge has been represented in computer based instruction is in terms of relationships (and relations between relationships). Variations on this approach also have a long history. In contemporary form, relational systems are
Joseph M. Scandura

often referred to as being model-based or constructivist in nature. This is not the place to dwell on details but rather just to present a more complete context.

Like Copernican Theory, both hierarchical and relational models work at higher levels of abstraction, and thereby simplify knowledge representation. They do so, however, at a cost. While varying degrees of support are presented in the literature, none are as precise or rigorous as in traditional ITS based on ACT-R. In effect, simple hierarchies and relational models in many ways play the same role as Copernican theory did with Earth centered Ptolemaic theory. Hierarchical and relational models are invariably simpler than ACT-R. At the same time, however, they do not account for the degree of detail offered by ACT-R.

In short, it is not a question of empirical grounding of ACT-R. As is clear from Ball’s (2011) summary (see Empirical Support in Appendix C), ACT-R has and continues to evolve as it attempts to account for increasingly refined data. This is precisely what Ptolemaic thinking required to preserve geocentric philosophy.

Increasing complications and the high cost of developing ITS based on ACT-R and other ITS-like systems (i.e., adaptive/individualized instructional systems) has had two major effects. First, focus has shifted to learning with technology versus using technology to facilitate learning. Technology is used as a cognitive tool as opposed to instructional aid. Second, contemporary work in adaptive or individualized learning systems is increasingly based on either relational models or BIG DATA (relationships between data bases of one sort or another). Constructivist theories play a major role in the former.

**Learning Analytics** is the latest iteration of the latter. Major reasons for this transition have been the difficulty and high cost of building traditional ITS and the relative ease of using technology as an aid. The use of technology as a tool has the disadvantage that many students simply do not have the wherewithal necessary to learn much of what they need without help. Constructivists prefer to use the term scaffolding for this purpose, but this simply renames what is effectively tutoring. Rather than throwing the baby out with the bath water, the challenge is to find more cost effective means of developing dynamically adaptive tutoring systems.

So-called adaptive learning systems based on learning analytics (e.g., Knewton & Dreambox Learning) substitute complex data analysis for theory. They offer a wide variety of empirically derived learning paths, and adjust these paths based on average student performance. TutorIT tutoring systems based on SLT are very different. Recently patented methods rest on a fundamentally different theoretical foundation – one that deals with individual knowledge rather than group statistics.

TutorIT interacts with students like a human tutor who is both intimately knowledgeable about the content to be taught and dedicated to making sure that each student masters that content to a pre-specified level of mastery. TutorIT
bases its decisions on the specific cognitive processes and decision making skills that must be learned for mastery in any given problem domain.

Learning analytics focuses on the way students learn. TutorIT focuses both on what the student must learn for success and when that information is needed. Oddly enough, the single most important conclusion drawn from my dissertation (Scandura, 1964a,b) was that when information is given is far more important than how that information is learned (e.g., by Exposition or Discovery). Moreover, decision making is deterministic in nature rather than probabilistic. TutorIT is designed to enable challenged students to catch up, and average students to move ahead faster, while still allowing gifted students to move at their own pace.

One additional point should be emphasized. I mentioned earlier but did not elaborate on the increasing attention being given to using the computer as a tool. This emphasis in no way diminishes the value of dynamically adaptive tutoring systems. Students do not come into the world knowing how to use computers to solve problems – irrespective of whether those skills involve writing letters or numerals, computations, simulations, or creating custom programs.

TutorIT and AuthorIT technologies as currently implemented are sufficiently broad to accomplish almost any desired web-based interactivity on pads or computers. Work that has been done makes it clear that AuthorIT can be used to develop and TutorIT to deliver dynamically adaptive tutoring systems for essentially any cognitive skill. Learning to use computers to solve various kinds of problems is no different than children learning how to do arithmetic or solve algebra word problems.

Moreover, recent advances makes it possible for AuthorIT (using SA in SLT) to represent requisite knowledge in arbitrary degrees of detail. Viability seems apparent from the sheer number and variety of dynamically adaptive tutoring systems a small team has been able to develop in a relatively short time.

A key question at this point in time is how difficult it will be to teach other subject matter experts to use AuthorIT to create TutorIT tutorials in their own areas of expertise.

7 CONCLUDING REMARKS

What does the above have to do with epicycles and their replacement by the Copernican solar centric model and later Kepler’s formalism’s? Both accounted for similar data, although Kepler did so far more simply and ultimately more precisely. This became apparent when matched against Tycho Brahe’s detailed data. Independent measurements beyond the solar system further confirmed the broader applicability of Kepler’s formalisms.
At this point in time, ACT-R is a highly refined theory backed by a substantial amount of empirical data. It rests on behavioral science’s traditional statistical foundation, wherein assumptions are made about individual behavior and test results are averaged over populations of subjects. Distrust in the complexities hidden in averages (despite, perhaps because of my early doctoral training in mathematical foundations and statistics), and motivated by the cohesiveness and simplicities of renascence physics, I came increasingly to focus on determinism as a better foundation for instructional theory. This focus is transparent in the title of my original article on SLT, *Deterministic theorizing in structural learning: Three levels of empiricism* (Scandura, 1971).

This focus developed gradually during my early days as a professor. It began with the realization in my earliest studies on problem solving in the “new math” that what is being learned by students and when is more important than how information is given and/or how the learning takes place. My students and I ran numerous experiments on rule learning at a time when S-R psychology held sway. These studies gradually led to the conclusion that the more precisely we could specify what needed to be learned for success, the less important the experimental work itself became. Nothing since that time has changed my mind.

In those early days, I used the terms “Structural” (cognitive task and/or domain) Analysis (SA) more or less interchangeably to distinguish SA from traditional task analysis. This was all before the label “Cognitive Task Analysis” (CTA) (as currently defined) was introduced. Emphasis in SA has always been on identifying with increasing precision what must be learned for success. Although discussion here is beyond the current scope, higher order rules (which operate on other rules) have played a central role in SLT from its inceptions (Scandura, 1971). SLT rules and higher order rules are not ingredients (e.g., productions) identified de novo by knowledge engineers. What must be learned for success comes directly from top-down domain analysis.

My early work on SA was motivated in part by Gagne’s work in the 1960s (building on Robert Miller’s, circa 1958) on Task Analysis. Gagne’s work focused on behavior. I was attracted to the idea generally, but differed in one fundamental respect. I felt that equal attention had to be given to what students had to learn in order to produce that behavior. I also found it hard to accept Gagne’s (e.g., 1965) characterization of facts, concepts, rules and problem solving as increasingly complex sets of associations. I proposed representing associations and concepts as special cases of (SLT) rules/principles (Scandura, 1967; 1970).

I had many discussions with Gagne at the time but failed to convince him that it was better to view rules as the general case with concepts and associations as special cases. Neither he nor most other contemporaries at the time (e.g., at LRDC) understood the fundamental difference between higher order rules that operate on and/or generate other rules and rules that are higher in a hierarchy (cf. Scandura, 1971, 2007). Mathematically speaking, the former correspond to functions that are
defined on and/or generate other functions. The latter correspond to rules that are
defined in terms of subordinate, typically sequences of simpler rules.

Another distinction that has rarely surfaced in the instructional design literature is that if there is any one way to accomplish a set of tasks, it necessarily follows that there must in principle be an indefinitely large number of other ways of doing the same thing. Viable theory must allow for multiple variations. Pask’s conversation theory (1974 with its focus on mutual understandings specifically addressed this issue and also influenced my thinking (cf. Pask, 1974; Durnin & Scandura, 1973). Compatibilities between the ways different individuals (e.g., teachers and learners) view the same content can have a fundamental effect on communication between teachers and learners.

Structural Learning Theory (SLT) rests on this foundation, combining knowledge representation, assessment of individual knowledge, acquisition of knowledge (learning) and interactions between learners and teachers (in principle between any two individuals, human or automated). SLT is a comprehensive theory of teaching and learning, an operationally defined theory that aims for cohesion and rigor along with a deterministic account of individual behavior. Rather than statistical prediction of averaged behavior, SLT focuses on the behavior of individual students (subjects) in specific situations. As in conversation theory, SLT also focuses on interactions between teachers and learners that lead to that behavior, albeit resting on a deterministic theoretical foundation that can compete equally as to precision with any other.

Given this background, it is instructive to revisit the Ptolemaic to Copernican to Kepler progression in parallel with ITS based on ACT-R to CBI based on Instructional Design (and/or Model based systems) to dynamically adaptive tutoring systems based on SLT. Earth centered Ptolemaic models offered a precise account of observable behavior, originally far better than the simpler Copernican Solar models. It took Kepler’s conversion to an elliptical model to achieve the same level of precision, albeit with a considerable reduction in explanatory simplicity.

In similar fashion, ITS research based on ACT-R has achieved considerable degrees of predictive precision (albeit stochastic precision). Empirical support for instructional systems based on traditional task analysis as used in instructional design and cognitive models is generally supportive but not nearly as precise. While the story has yet to be told in full, there is no reason to assume that deterministic accounts made possible by SLT cannot provide equal precision. Indeed, since SLT itself is deterministic rather than probabilistic in nature, SLT offers the potential of achieving even greater predictive precision of individual behavior as opposed to averages.

As final note, I recently returned from the first TICL presidential round table (at AERA in San Francisco) featuring Dick Clark and myself on bottom-up versus top-down approaches to knowledge representation (CTA/SA). While supporting a bottom-up approach, Clark has dealt with more complex domains than those
traditionally dealt with in ITS. In the process he has found various techniques for identifying the 70% he reports missing in standard CTA. Some refer to that missing 70% as “tacit” or implicit knowledge. (I prefer the terms higher level and/or higher order knowledge.)

Clark illustrated his approach with the domain of patent examination – an especially timely subject because I had just learned that a patent application covering AuthorIT and TutorIT had passed final muster. I took Dick’s sample analysis of the patent examination process as a starting point. I then showed how AuthorIT could be used to represent what needed to be learned (about patent examination), as precisely as one wanted – systematically from the top down. I also was able to pass the resulting knowledge representation to TutorIT showing how trainees could be tutored in the requisite knowledge. This was all very preliminary, of course. We did this in only two days.

This simple exercise, however, illustrates some important points. One can supplement low level knowledge obtained in standard CTA as Dick (and other experienced folks like Jeroen van Merrienboer who joined in) suggests by explicitly attending to “tacit” knowledge. Using SA, one can achieve the same level of completeness and precision. In both cases, bottom up supplemented with tacit knowledge and arbitrarily precise hierarchical representations resulting from top-down SA can be converted into dynamically adaptive tutoring systems.

The major difference is in the amount of work and testing required. Standard bottom-up approaches are costly to develop and costly to adjust. Converting the results of CTA into a dynamically adaptive tutoring system, takes a lot of work – a lot of programming, a lot of testing and a lot of revision. AuthorIT and TutorIT, on the other hand, make all pedagogical decisions – what to test, what to teach and when – all automatically. The results of SA are fed directly to TutorIT. TutorIT in turn delivers the content to the student automatically, dynamically determining what the student knows at each point in time and giving the student what he or she needs to progress, when that information is needed.

TutorIT ensures that every student who enters with pre-specified prerequisites, and who completes a given TutorIT tutorial will necessarily have demonstrated mastery of the content to the pre-specified level. It is not a question of mastery as such, but whether a student will be motivated to complete any given tutorial and how long it will take. These remaining factors can all be addressed incrementally as desired – all without fundamental change of the tutoring system itself. AuthorIT and TutorIT collectively make this possible for the first time.

Given these now indisputable facts, it will be interesting to see how historical parallels with respect to ACT-R and SLT will play out.
ESSENTIAL REFERENCES


OTHER REFERENCES


Joseph M. Scandura


Ball, J.T. Explorations in ACT-R Based Cognitive Modeling – Chunks, Inheritance, Production Matching and Memory in Language Analysis. Advances in Cognitive Systems: Papers from the 2011 AAAI Fall Symposium (FS-11-01)


Ball, Jerry T. Explorations in ACT-R Based Language Analysis – Memory Chunk Activation, Retrieval and Verification without Inhibition, Air Force Research Laboratory, Wright-Patterson Air Force Base, jerry.ball.@wpafb.af.mil


Miller, G. A. The magic number seven, plus or minus two. Psychological Review, 1956, 63, 81–97.


Scandura, J. M. Problem solving in schools and beyond: Transitions from the naive to the neophyte to the master. *Educational Psychologist*, 1981, 16, 139–150.


Scandura, J. M. Structural (cognitive task) analysis: an integrated approach to software design and programming. *Journal of Structural Learning and Intelligent Systems*, 2001, 14, 4, 433–458. (c)


Winne, P. et al, Supporting self-regulated learning with gStudy software: the learning kit project.

RELEVANT PATENTS

APPENDIX A: REPLACING PTOLEMAIC COSMOLOGY WITH COPERNICAN RATIONALE AND KEPLER’S FORMALISMS (DERIVED FROM WIKIPEDIA)

The Earth-centered Universe of Aristotle and Ptolemy held sway on Western thinking for almost 2000 years. Then, in the 16th century a new idea was proposed by the Polish astronomer Nicolai Copernicus (1473–1543).

The Heliocentric System

In a book called *On the Revolutions of the Heavenly Bodies* (that was published as Copernicus lay on his deathbed), Copernicus proposed that the Sun, not the Earth, was the center of the Solar System. Such a model is called a *heliocentric system*. The ordering of the planets known to Copernicus in this new system is illustrated in the following figure, which we recognize as the modern ordering of those planets.

![The Copernican Universe](image)

In this new ordering the Earth is just another planet (the third outward from the Sun), and the Moon is in orbit around the Earth, not the Sun. The stars are distant
objects that do not revolve around the Sun. Instead, the Earth is assumed to rotate once in 24 hours, causing the stars to appear to revolve around the Earth in the opposite direction.

**Retrograde Motion and Varying Brightness of the Planets**

The Copernican system by banishing the idea that the Earth was the center of the Solar System, immediately led to a simple explanation of both the varying brightness of the planets and retrograde motion:

1. The planets in such a system naturally vary in brightness because they are not always the same distance from the Earth.
2. The retrograde motion could be explained in terms of geometry and a faster motion for planets with smaller orbits.

See original Wikipedia for animation of Retrograde motion in the Copernican System.

A similar construction can be made to illustrate retrograde motion for a planet inside the orbit of the Earth.

**Copernicus and the Need for Epicycles**

There is a common misconception that the Copernican model did away with the need for epicycles. This is not true. Copernicus was able to rid himself of the long-held notion that the Earth was the center of the Solar system, but he did not question the assumption of uniform circular motion. Thus, in the Copernican model the Sun was at the center, but the planets still executed uniform circular motion about it. As we shall see later, the orbits of the planets are not circles, they are actually ellipses. As a consequence, the Copernican model, with its assumption of uniform circular motion, still could not explain all the details of planetary motion on the celestial sphere without epicycles. The difference was that the Copernican system required many fewer epicycles than the Ptolemaic system because it moved the Sun to the center.

**The Copernican Revolution**

We noted earlier that 3 incorrect ideas held back the development of modern astronomy from the time of Aristotle until the 16th and 17th centuries: (1) the assumption that the Earth was the center of the Universe, (2) the assumption of uniform circular motion in the heavens, and (3) the assumption that objects in the heavens were made from a perfect, unchanging substance not found on the Earth.
Copernicus challenged assumption 1, but not assumption 2. We may also note that the Copernican model implicitly questions the third tenet that the objects in the sky were made of special unchanging stuff. Since the Earth is just another planet, there will eventually be a natural progression to the idea that the planets are made from the same stuff that we find on the Earth.

Copernicus was an unlikely revolutionary. It is believed by many that his book was only published at the end of his life because he feared ridicule and disfavor: by his peers and by the Church, which had elevated the ideas of Aristotle to the level of religious dogma. However, this reluctant revolutionary set in motion a chain of events that would eventually (long after his lifetime) produce the greatest revolution in thinking that Western civilization has seen. His ideas remained rather obscure for about 100 years after his death. But, in the 17th century the work of Kepler, Galileo, and Newton would build on the heliocentric Universe of Copernicus and produce the revolution that would sweep away completely the ideas of Aristotle and replace them with the modern view of astronomy and natural science. This sequence is commonly called the *Copernican Revolution*.

**Been There, Done That: Aristarchus of Samos**

The idea of Copernicus was not really new! A sun-centered Solar System had been proposed as early as about 200 B.C. by Aristarchus of Samos (Samos is an island off the coast of what is now Turkey). However, it did not survive long under the weight of Aristotle’s influence and “common sense”:

1. If the Earth actually spun on an axis (as required in a heliocentric system to explain the diurnal motion of the sky), why didn’t objects fly off the spinning Earth?
2. If the Earth was in motion around the sun, why didn’t it leave behind the birds flying in the air?

3. If the Earth were actually on an orbit around the sun, why wasn’t a parallax effect observed? That is, as illustrated in the adjacent figure, stars should appear to change their position with the respect to the other background stars as the Earth moved about its orbit, because of viewing them from a different perspective (just as viewing an object first with one eye, and then the other, causes the apparent position of the object to change with respect to the background).

The first two objections were not valid because they represent an inadequate understanding of the physics of motion that would only be corrected in the 17th century. The third objection is valid, but failed to account for what we now know to be the enormous distances to the stars. As illustrated in the following figure, the amount of parallax decreases with distance.

The parallax effect is there, but it is very small because the stars are so far away that their parallax can only be observed with very precise instruments. Indeed, the parallax of stars was not measured conclusively until the year 1838. Thus, the heliocentric idea of Aristarchus was quickly forgotten and Western thought stagnated for almost 2000 years as it waited for Copernicus to revive the heliocentric theory.
APPENDIX B: OVERVIEW OF ACT-R
(DERIVED FROM WIKIPEDIA)

ACT-R has been inspired by the work of Allen Newell, and especially by his
life-long championing the idea of unified theories as the only way to truly uncover
the underpinnings of cognition, which ironically has long paralleled my own
deterministic goals. Indeed, Newell & Simon’s book on “Human Problem Solving”
and mine on “Structural Learning I. Theory and Research” both came out in late
1972/3 and were the focus of a joint review in Contemporary Psychology.[1] In
fact, John Anderson usually credits Allen Newell as the major source of influence
over his own theory.

In recent years, ACT-R has also been extended to make quantitative predic-
tions of patterns of activation in the brain, as detected in experiments with fMRI.
In particular, ACT-R has been augmented to predict the shape and time-course of
the BOLD response of several brain areas, including the hand and mouth areas in
the motor cortex, the left prefrontal cortex, the anterior cingulate cortex, and the
basal ganglia.

ACT-R’s most important assumption is that human knowledge can be divided
into two irreducible kinds of representations: declarative and procedural.

Within the ACT-R code, declarative knowledge is represented in form of
chunks, i.e. vector representations of individual properties, each of them acces-
sible from a labelled slot.

Chunks are held and made accessible through buffers, which are the front-end
of what are modules, i.e. specialized and largely independent brain structures.

There are two types of modules:

• Perceptual-motor modules, which take care of the interface with the real
world (i.e., with a simulation of the real world). The most well-developed
perceptual-motor modules in ACT-R are the visual and the manual modules.

• Memory modules. There are two kinds of memory modules in ACT-R:
  • Declarative memory, consisting of facts such as Washington, D.C. is the
capital of United States, France is a country in Europe, or 2 + 3 = 5
  • Procedural memory, made of productions. Productions represent knowledge
about how we do things: for instance, knowledge about how to type the letter
“Q” on a keyboard, about how to drive, or about how to perform addition.

All the modules can only be accessed through their buffers. The contents of the
buffers at a given moment in time represents the state of ACT-R at that moment.
The only exception to this rule is the procedural module, which stores and applies
procedural knowledge. It does not have an accessible buffer and is actually used to access other module’s contents.

Procedural knowledge is represented in form of *productions*. The term “production” reflects the actual implementation of ACT-R as a production system, but, in fact, a production is mainly a formal notation to specify the information flow from cortical areas (i.e. the buffers) to the basal ganglia, and back to the cortex.

At each moment, an internal pattern matcher searches for a production that matches the current state of the buffers. Only one such production can be executed at a given moment. That production, when executed, can modify the buffers and thus change the state of the system. Thus, in ACT-R, cognition unfolds as a succession of production firings.

Members of the ACT-R community, including its developers, prefer to think of ACT-R as a general framework that specifies how the brain is organized, and how its organization gives birth to what is perceived (and, in cognitive psychology, investigated) as mind, going beyond the traditional symbolic/connectionist debate. None of this, naturally, argues against the classification of ACT-R as symbolic system, because all symbolic approaches to cognition aim to describe the mind, as a product of brain function, using a certain class of entities and systems to achieve that goal.

A common misunderstanding suggests that ACT-R may not be a symbolic system because it attempts to characterize brain function. This is incorrect on two counts: First, because all approaches to computational modeling of cognition, symbolic or otherwise, must in some respect characterize brain function, because the mind is brain function. And second, because all such approaches, including connectionist approaches, attempt to characterize the mind at a cognitive level of description and not at the neural level, because it is only at the cognitive level that important generalizations can be retained.\[3\]

Further misunderstandings arise because of the associative character of certain ACT-R properties, such as chunks spreading activation to each other, or chunks and productions carrying quantitative properties relevant to their selection. None of these properties counter the fundamental nature of these entities as symbolic, regardless of their role in unit selection and, ultimately, in computation.

ACT-R has been often adopted as the foundation for *cognitive tutors*.\[32\][33] These systems use an internal ACT-R model to mimic the behavior of a student and personalize his/her instructions and curriculum, trying to “guess” the difficulties that students may have and provide focused help.

Such “Cognitive Tutors” are being used as a platform for research on learning and cognitive modeling as part of the Pittsburgh Science of Learning Center.
of the most successful applications, like the Cognitive Tutor for Mathematics, are used in thousands of schools across the United States.

**Current developments: 1998–present**

After the release of ACT-R 4.0, John Anderson became more and more interested in the underlying neural plausibility of his life-time theory, and began to use brain imaging techniques pursuing his own goal of understanding the computational underpinnings of human mind.

The necessity of accounting for brain localization pushed for a major revision of the theory. ACT-R 5.0 introduced the concept of modules, specialized sets of procedural and declarative representations that could be mapped to known brain systems. In addition, the interaction between procedural and declarative knowledge was mediated by newly introduced buffers, specialized structures for holding temporarily active information (see the section above). Buffers were thought to reflect cortical activity, and a subsequent series of studies later confirmed that activations in cortical regions could be successfully related to computational operations over buffers.

A new version of the code, completely rewritten, was presented in 2005 as ACT-R 6.0. It also included significant improvements in the ACT-R coding language.

**Empirical Support**

A recent paper by Ball (2011) explores the benefits and challenges of using the ACT-R cognitive architecture in the development of a large scale, functional, cognitively motivated language analysis model. The paper focuses on chunks, inheritance, production matching and memory, proposing extensions to ACT-R to support multiple inheritance and suggesting a mapping from the focus of attention, working memory and long-term memory to ACT-R buffers and declarative memory (DM).

ACT-R is a hybrid symbolic, subsymbolic (or probabilistic) architecture which combines parallel, probabilistic mechanisms for declarative memory (DM) chunk activation and selection (i.e. retrieval), and a parallel, utility based production matching and selection mechanism, with a serial production execution mechanism. The production system is the central component of ACT-R. It interfaces to DM and other cognitive/perceptual modules (e.g. motor module, visual module) via a collection of module specific buffers which contain chunks that constitute the current context for production matching. Buffers are restricted to containing a single chunk at a time. The production with the highest utility which matches the current context is selected and executed. Execution of a production may effect
an action resulting in a change to the current context (e.g. via retrieval of a DM chunk, or a shift in attention to a new visual object). The changed context determines which production next matches and executes.

To the extent that there is empirical support for constructs like the episodic buffer and LTWM, and, more generally, to the extent that there is empirical motivation for a larger working memory than provided by the built-in ACT-R buffers, the collection of buffers used in the language analysis model is also supported. In a recent presentation by John Anderson (June 2011), he introduced a new collection of buffers to support metacognition in ACT-R. Metacognition occurs during the completion of complex tasks and encompasses processes like reflection, high-level reasoning and theory of mind. Anderson provided fMRI evidence for these buffers based on the activation of several distinct brain regions during the performance of complex algebraic tasks. Anderson also noted that the performance of tasks which require a mapping from learned techniques for completing algebraic equations, to novel, but isomorphic, ways of representing algebraic tasks requires simultaneous maintenance of at least two chunks to perform the mapping from learned to novel representation. The introduction of these new buffers and the recognition of the need to maintain multiple chunks for comparison and analogy extends the capability of ACT-R for modeling complex tasks, and begins to address some of the large areas of the brain for which ACT-R currently has little to say (including much of the pre-frontal cortex).

ACT-R’s constraints on production matching and the limits of single inheritance, combined with assumptions about chunk size and local access to chunk contents have created opportunities and challenges in the development of a large scale functional language analysis model. The problems can be allayed to some extent by the addition of buffers to retain the partial products of language analysis and the creation of additional productions where multiple-inheritance is needed, but not supported. Empirical support for the addition of these buffers is provided via association with Baddeley’s episodic buffer and, to some extent, with Ericsson & Kintsch’s LTWM. Also see Anderson, J.R., Fincham, J. M., Qin, Y., & Stocco, A. (2008). A central circuit of the mind. Trends in Cognitive Science, 12(4), 136–143.
# Appendix C: Historical Parallels

<table>
<thead>
<tr>
<th>Ptolemaic Earth Centered Theory</th>
<th>ACT-R &amp; ITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movement of planets best viewed from geo-centric perspective</td>
<td>Tutoring process best viewed from a brain centric perspective</td>
</tr>
<tr>
<td>Supported by centuries of increasingly precise data</td>
<td>Supported by large amounts of increasingly precise data collected over decades</td>
</tr>
<tr>
<td>Increasing complexity of supporting theory involving increasing numbers of epicycles and offsets</td>
<td>Increasing complexity of supporting theory involving increasing numbers of perceptual, goal, declarative memory modules in buffers accessed by production systems</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Copernican Sun Centered Theories</th>
<th>Instructional Design &amp;/or Model-based theories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movement of planets best viewed from solar perspective</td>
<td>Tutoring process best viewed from task analysis &amp;/or relational model perspective</td>
</tr>
<tr>
<td>Approximate support for available data</td>
<td>Approximate support for available data</td>
</tr>
<tr>
<td>Circular movement considerably reduced complexity of explanation</td>
<td>Hierarchical and relational models reduced complexity of explanation and development</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Kepler Sun Centered Theory</th>
<th>Structural Learning Theory (SLT) &amp; TutorIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movement of planets best viewed from solar perspective</td>
<td>Tutoring process best viewed from a knowledge centric perspective</td>
</tr>
<tr>
<td>Supported precise data collected over decades, including Tycho-Brahe</td>
<td>Supported by decades of generally approximate data plus foundational deterministic data</td>
</tr>
<tr>
<td>Elliptical movement both provided precise account of data AND dramatically reduced complexity of explanation</td>
<td>Offers deterministic account of data AND dramatic reduction in complexity of explanation and development efficiencies (eliminates need for programming pedagogical decision making)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ACT-R &amp; ITS</th>
<th>SLT &amp; TutorIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge is procedural or declarative</td>
<td>All knowledge is BOTH structural/declarative and procedural</td>
</tr>
<tr>
<td>Strengthening: high level to compiled code</td>
<td>Strengthening: procedural to data structure</td>
</tr>
<tr>
<td>Effective capacity goes up with practice because of compilation</td>
<td>Effective capacity goes up with practice because of procedural processing is replaced by direct perception/decoding</td>
</tr>
</tbody>
</table>