THE TECHNOLOGY ACCEPTANCE MODEL (TAM) 
(Davis, 1989) have been rigorously evaluated, 
modified, and enhanced in a series of research 
studies during the last 2 decades (Bhattacher- 
jee & Premkumar, 2004; Davis, 1989, 1993; 
Venkatesh, 2000; Venkatesh & Davis, 1996, 
2000; Venkatesh, Morris, Davis, & Davis, 
2003). The perceived ease of use and the per- 
ceived usefulness measures have been used 
to explain and/or predict the user acceptance 
of various IT artifacts such as digital libraries 
(Hong, Thong, Wong, & Tam, 2002), shopping 
on the Web (Shih, 2004), the World Wide Web 
(Fenech, 1998), and expert system advice (Ye 
& Johnson, 1995).

With some modification, the perceived ease 
of use and the perceived usefulness measures 
have also been used to assess the usability 
of models as well as modeling techniques 
in systems analysis and design and database 
management (Batra, Hoffer, & Bostrom, 1990; 
Hardgrave & Dalal, 1995; Kim & March, 1995; 
Topi & Ramesh, 2001). However, the perceived 
ease-of-use measure only tells us if a modeling 
technique is easy or difficult for a given kind 
of analysts (e.g., for novice, experienced, or 
expert); it does not suggest ways by which the 
technique can be improved or reengineered.

There is good research to support that a 
difficult modeling technique will be rejected 
by the analyst. However, an easy modeling 
technique may not help if learning does not take 
place. Learning does involve a certain amount 
of cognitive load for schema construction 
and automation (van Merriënboer & Sweller, 
2005). Yet, the perceived ease-of-use measure 
ofers little guidance on the effectiveness of a 
modeling technique in the vast range between 
the very easy and very difficult.

The computer self-efficacy (CSE) was first 
introduced to compare the effects of alterna- 
tive training methods on the performance in 
computer software training (Gist, Schwoerer, 
& Rosen, 1989). The computer self-efficacy is 
defined as “judgment of one’s capability to use 
a computer” (Compeau & Higgins, 1995). It is 
the computing domain-specific extension of the 
self-efficacy construct derived from Bandura’s 
(1982) social cognitive theory. The self-efficacy 
concept is based on the premise that one’s belief 
in one’s cognitive capabilities influences one’s 
behavior in performing a task and vice versa 
(Marakas, Yi, & Johnson, 1998). The Compeau 
and Higgins’s (1995) influential paper led to
a line of research (Agarwal, Sambamurthy, & Stair, 2000; Compeau, Higgins, & Huff, 1999; Munro, Huff, Marcolin, & Compeau, 1997), and CSE is well accepted as a predictor of user performance.

However, how does one gain CSE? It is logical to assume that learning is essential if self-efficacy is to be attained; an analyst cannot be confident in her capability to use a computer, a software, or a modeling technique if she has not learned the associated materials. Just as self-efficacy facilitates learning, self-efficacy can be a result of learning given that the questionnaires on self-efficacy are based on items that assess learning, experience, and proficiency. For example, assume that a modeling technique is intended to train novice analysts on developing entity relationship (ER) diagrams. An ER diagram specific self-efficacy questionnaire will directly or indirectly test the amount of learning rendered during a training session, or from experience.

It is, thus, important to identify one or more relevant learning-oriented theories that can guide research on devising information systems modeling techniques. Siau (1999) has proposed the use of cognitive psychology as a reference discipline for information modeling, and method engineering, which is “the process of designing, constructing, and adapting modeling methods for the development of information systems.” The term “modeling technique” used in this article is similar to the use of the term “method engineering” by Siau (1999), who notes that most modeling constructs are employed based on common sense, superficial observation, and intuition of researchers and practitioners. This calls for a scientific and theory-based to devise modeling techniques. This preface considers the cognitive load theory (CLT) as a promising vehicle to provide valuable guidelines in creating instructional materials for learning modeling techniques. It is implicit that the techniques that result by applying CLT will need to be empirically evaluated in laboratory and field settings.

THE RELEVANCE OF COGNITIVE LOAD THEORY (CLT)

The focus of this preface is on the use of CLT in learning an information systems modeling technique. In general terms, we expect a good modeling technique to be effective and efficient; however, effectiveness and efficiency are measures that can be assessed only after the technique has been developed. We need to identify the underlying factors that influence the construction of the technique so that it ends up as effective and efficient. Further, the modeling technique should facilitate learning by employing the limited cognitive resources of a novice analyst. The CLT provides a sound theoretical basis for constructing an effective and efficient technique.

CLT assumes a limited working memory that stores about seven elements (Miller, 1956) but operates on just two to four elements (van Merriënboer & Sweller, 2005). The theory emphasizes that this working memory capacity applies only to novel information obtained through sensory memory; working memory has no known limitations when dealing with information retrieved from long-term memory (Ericsson & Kintsch, 1995). Human expertise comes from knowledge stored in these schemata, not from an ability to engage in reasoning with many elements that have not been organized in long-term memory. Human working memory simply is not able to process many elements. Expertise develops as learners mindfully combine simple ideas into more complex ones and store them as an element in the long-term memory.

The Cognitive Load Theory (CLT) postulates that the cognitive load introduced by the task or the problem-solving strategy may potentially hinder learning (Sweller, 1988). Any problem-solving strategy or task that imposes more cognitive load than the working memory can handle, more or less seven pieces of information, eventually overwhelm the learner and affect the performance. For example, sequence diagram modeling in object-oriented analysis is one such task that involves a high number of
interacting elements (various kinds of objects, rules of interactions, and the messages) that often require concurrent processing in the working memory. For a novice analyst, sequence diagram modeling can cause cognitive load.

Other examples can be more insidious. One may consider the entity relationship (ER) diagram as having only three main constructs: entity, relationship, and attribute. However, as the number of entities increase linearly, the number of relationships expand in a combinatorial fashion, but only a small number are admissible. Once a task has the combinatorial expansion problem, the working memory is unable to cope with the complexity (van Merriënboer & Sweller, 2005). Thus, ER modeling problems are hard despite the innocuous façade of just three essential constructs.

The vast majority of the research on CLT is concerned with reducing the cognitive load by manipulating the design of the instructional materials (Chandler & Sweller, 1991; Kalyuga, Chandler, & Sweller, 1998, 1999; Kirschner, 2002; Paas, Renkl, & Sweller, 2003; van Merriënboer, Schuurman, de Croock, & Paas, 2002). Thus, CLT can be a key theory in the design of information systems modeling techniques. CLT researchers have identified a number of effects that can reduce the cognitive load associated with the instructional design, including goal-free effect, worked example effect, completion problem effect, split-attention effect, modality effect, and redundancy effect (Sweller, van Merriënboer, & Paas, 1998; van Merriënboer et al., 2002), most of which are concerned with the presentation of the materials. For example, the graphical nature of Unified Modeling Language (UML) and data flow diagrams provides a presentation mechanism that can mitigate cognitive load (Siau & Tian, 2009). However, CLT can suggest ways to further reduce the load by applying and evaluating strategies that have worked in other domains.

Recent studies have attempted to reduce the cognitive load inherent to the task by designing instructions that can reduce the number of interacting elements involved in a task (Bannert, 2002; van Merriënboer & Sweller, 2005). One way of accomplishing this is to break down the complex task into meaningful parts or modules that can be tackled separately from each other (Gerjets, Scheiter, & Catrambone, 2004). Thus, it is natural to first break down the modeling of a sequence diagram into the object aspect and the message aspect, but the more detailed breakdown is not clear and needs to be devised.

A reduction in the cognitive load by suitable presentation and by breaking down a complex task can free up cognitive resources that can be used for learning. The learning aspect can be enhanced by providing mechanisms for schema construction (van Merriënboer et al., 2002).

**The Three Types of Cognitive Loads**

Managing the cognitive load when devising an instructional technique is not simple, however. Literature on instructional techniques reveals that a technique that is suitable for a given analyst needs to challenge her to cause a certain amount of cognitive load (van Merriënboer & Sweller, 2005). However, what constitutes an appropriate amount may be an empirical question. Since there is a limit to the overall cognitive load that can be managed by a novice analyst, we need to understand the kinds of cognitive loads when devising experimental studies to evaluate a modeling technique.

There are three kinds of cognitive loads: intrinsic, extraneous, and germane (van Merriënboer & Sweller, 2005). Intrinsic load refers to the inherent load in the instruction, and arises because of the interactivity among the elements in a task. Extraneous load is generated by the manner in which the information is presented. It is the load that is not necessary for learning and that can be altered by instructional intervention. Germane load is load devoted to the processing, construction, and automation of schemata. Since the overall cognitive load needs to be controlled, space for germane load can exist only if intrinsic and extraneous loads are controlled.

The intrinsic load of several systems analysis and design tasks is generally high as data, processes, and actors interact with each
other. Breaking down the task can reduce the intrinsic load. However, the individual sub-tasks needs to be brought together eventually, and indiscriminate breakdown may actually increase cognitive load (Pollock, Chandler, & Sweller, 2002). The SA&D techniques have done well to adopt graphical notations in the modeling techniques to reduce extraneous load, but research can lead to further improvements. For example, it has been shown (Masri, Parker, & Gemino, 2008a) that the extraneous load in the ER model can be further reduced by using icons. Siau and Tian (2009) have employed semiotics to show that the use of iconic signs as UML graphical notations leads to better representations that with the use of symbolic notations.

Consider how the three cognitive loads can be managed in the domain of data modeling, which has a high intrinsic load because of the large number of relationships possible among a small number of entities. A number of studies (e.g., Batra et al., 1990; Bock & Ryan, 1993; Jarvenpaa & Machesky, 1989; Juhn & Naumann, 1985; Palvia, Liao, & To, 1992) have indicated that the entity relationship (ER) model, the graphical form of the model resulted in better performance as compared the text-based relational model. Thus, the graphical ER model reduces the extraneous load in modeling relationships.

However, the intrinsic load remains high because of the combinatorial expansion of relationships as the number of entities increased. Theorey, Wei, Bolton, and Koenig (1989) argue that a database model can be considered as consisting of interrelated clusters, which mitigates the complexity of data models by providing layers of abstraction. Since a typical cluster has four to eight entities, the intrinsic load is confined within the cluster. Clustering facilitates a chunking mechanism that limits the intrinsic load.

Yet, for a novice analyst, this load can still be significant even with five or six entities. To manage this load, rules, heuristics, and patterns need to be provided (Batra, 2005; Batra & Zanakis, 1994; Geerts & McCarthy, 2002; McCarthy, 2003). However, learning the rules, heuristics, and patterns increases the cognitive load. This load is the germane load. When the extraneous load has been reduced, and the intrinsic load has been managed, there can be room in the working memory to take up the germane load to learn and store schemas that can later be retrieved from the long-term memory.

One may consider cognitive load as a stress on the human system, and thus, the natural tendency is to reduce this load. However, it can be argued that a certain load is necessary for learning to take place. When learning a modeling technique with a significant degree of complexity, a small cognitive load may likely imply that the germane load is not invoked to facilitate learning. Thus, a perceived ease-of-use measure may not be useful in this context. However, a very high cognitive load will invariably lead to rejection of the technique. In between the two extremes, somewhere, there is an optimal mix of the three kind of loads, which depends on the expertise level of the analyst.

Instructional methods that work well for novice learners may have neutral or even negative effects when expertise increases (van Merriënboer et al., 2005). It is, therefore, necessary to determine the level of difficulty and the actual problems encountered by a learner before devising a modeling technique. This can be done by assessing the expertise and the problems encountered by using questionnaires and by interviewing the analysts after they have attempted a sample problem.

COGNITIVE LOAD AND SA&D METHODOLOGIES

A technique is generally focused on one aspect of a methodology. Thus, an object-oriented methodology can involve use case diagrams, use case descriptions, domain modeling, sequence diagrams, and class diagrams to cover different stages of systems development. Each aspect of a methodology may require its own technique. The overall cognitive load for developing a system needs to consider the cognitive load of individual techniques comprising the methodology as well as the load incurred when transitioning from one diagram to another.
Thus, a methodology may be very effective in conducting analysis, but may not provide an effective technique to move into design. Another methodology may provide a set of disparate diagrams that can be used as needed by the analysts and designers for specific tasks, but that do not provide an integrated view or show how one diagram can be linked to another. Thus, the cognitive load may be low for individual techniques, but when the methodology is considered as a whole, the cognitive load is substantially high. These issues are quite real as we have two major methodology camps today—structured and object-oriented—and both methodologies suffer from a high cognitive load, although for different reasons.

In structured analysis and design, the data flow diagram (DFD) is considered the key analysis diagram (Gane & Sarson, 1977). Techniques that employ the DFD construct are strong on functional decomposition, which reduces intrinsic load. Further, the graphical feature of the DFD reduces the extraneous load. However, the structured design, based on structured charts, is removed from the deliverable of the analysis stage. Thus, the intrinsic load for analysis may be low, but integrating analysis and design can create significant intrinsic load. Further, in the structured approach, the design is not linked to coding, thereby exacerbating the cognitive load.

This shortcoming and the advent of object-oriented programming have led to the object-oriented approach, which is represented by the Unified Modeling Language (UML), which provides about a dozen diagrams. However, UML, too, has usability issues (Siau & Loo, 2006). Analysts are free to pick and choose diagrams relevant to their application, but precise modeling techniques do not exist. In practice, however, only three diagrams are commonly used (Dobing & Parsons, 2008)—the class diagram, the use case diagram (along with the written use cases), and the sequence diagram. UML diagrams are sometimes confused with modeling techniques. The sequence diagram, for example, provides the terminology, constructs, and essential rules. However, it would be a stretch to call the sequence diagram as a modeling technique. The diagram only provides the “what” aspect, while a technique provides the “how” aspect. For a novice analyst, the lack of a technique can result in a high cognitive load. Further, although use cases are employed in the object-oriented approach, the lack of a technique that facilitates systems decomposition is a major shortcoming, which could result in increased intrinsic load.

If the structured and the object-oriented approaches have shortcomings from a CLT perspective, what can future research do to solve the issue? The best principles from the two approaches can be combined, or an integrated methodology can be proposed. A few such attempts have been made. For example, Shoval and Kabeli (2001) have proposed the Functional and Object-Oriented Methodology (FOOM), an integrated methodology for information systems analysis and design. The FOOM methodology skillfully blends the ER diagram, the DFD, and certain aspects of UML. Dori (2002) has proposed a similar methodology called Object Process Methodology (OPM), which is based on minimizing the number of diagrams by integrating the data and functional modeling. A recent modeling technique (Masri, Gemino, & Parker, 2008b) specifically employs CLT to combine UML views. Although FOOM and OPM have been experimentally compared (Kabeli & Shoval, 2005) and have been found to be fairly similar, the comparison among the newer and conventional techniques raises interesting research questions. Further, given that the sequence diagram brings objects and messages together, a variant may provide an integrated diagram that is consistent with the object-oriented approach and that provides a basis for comparison. Based on the cognitive load theory, the integrated “techniques” should perform better.

Cognitive Load Theory and Cognitive Complexity

In the instructional literature, the link between the cognitive load theory and cognitive complexity is not well established. Implicitly, they seem to refer to the same problem, given that
cognitive complexity add neural load to learning any task (Reeves, 1999). Cognitive Complexity refers to external elements that contribute directly to our neural load, thereby reducing our capacity to think clearly and understand: cognitive complexity in the design context is the sum of those external factors that make things hard to see and use, hard to grasp, and hard to understand (Reeves, 1999). One way to distinguish cognitive load from cognitive complexity is that while the cognitive load is internally felt by the analyst, the cognitive complexity refers to external factors that cause cognitive load. The extant literature does not provide a good link between cognitive load and cognitive complexity, although it appears logical that a systematic analysis of cognitive complexity can provide strategies to reduce cognitive load.

Cognitive complexity factors are based on five sources: metasocial forces, information overload, complex problems, system complexity, and incoherent design. Reeves (1999) lists a total of 44 complexity causing factors under these five categories. For example, high number of variables is one of the complexifying factors. Research on structural complexity of UML (Siau & Cao, 2001; Siau, Erickson, & Lee, 2005), for example, employs the number of constructs in a given UML diagram to assess the complexity. The number of constructs count is similar to the number of variables count as a predictor of complexity; here, the structural complexity results in cognitive complexity, which in turn, cause cognitive load.

The 44 factors listed in Reeves (1999) have some redundancy; however, a statistical analysis should reveal a fair number of underlying complexity factors. These factors can provide new insights when designing a modeling technique. Consider a complexifying factor—disorder—from the information overload source. The simplifying counterpart—order—can reduce the extraneous load. A complexifying factor from systems complexity—interactive subsystems—can increase the intrinsic load. By decomposing a problem into manageable chunks, the intrinsic load can be decreased. Finally, a complexifying factor from problem solving—expertiserequired—refersto schemas such as rules, heuristics, and patterns that are required to solve a problem. When these schemas are rendered at the novice analyst level, these can increase the germane load, which is the useful kind of load that enhances learning. Note that the cognitive load theory does not require that the load be minimal. In order for learning to take place in a difficult situation, a certain amount of cognitive load is essential. When this load is too little or too much, effective learning does not take place.

**CONCLUSION**

It should be feasible to apply the cognitive complexity factors in a variety of learning situations such as for modeling the sequence diagram (VanderMeer & Dutta, 2009), the data flow diagram, workflow diagrams, data modeling diagrams, the activity diagram, Petri nets, the FOOM diagram, and the OPM diagram. As Siau (1999) points out, theoretical foundations and empirical studies provide a sound mix for method engineering. CLT and cognitive complexity are candidates to provide the theoretical foundation when devising information systems modeling techniques.

**REFERENCES**


Dinesh Batra is a Knight-Ridder research professor at the Department of Decision Sciences and Information Systems in the College of Business Administration at the Florida International University. Dr. Batra has published articles in journals such as Management Science, Communication of the ACM, Journal of MIS, Journal of Database Management, International Journal of Human Computer Studies, Data Base, European Journal of Information Systems, Communications of the AIS, Decision Support Systems, Requirements Engineering, Computers and OR, Information Systems Management, and Information & Management. His research interests focus on systems analysis and design methods, usability issues in systems and databases, and distributed development. He is currently an associate editor in the Journal of Database Management, and Communications of the AIS, and senior editor in the Information Systems Management. He is a co-author of the book Object-Oriented Systems Analysis and Design. He has served as a President of the AIS Special Interest Group on Systems Analysis & Design (SIGSAND).