

# Lessons from Cognitive Science for Computer Science Education

Matt Bower  
Macquarie University

## Abstract

This paper articulates stages of cognition that underpin learning, as identified by the fields of neuroscience and cognitive psychology. The stages are presented in the context of developing Computer Science thinking skills. Then strategies from educational literature (both general and computer science specific) are mapped to each of these stages of cognition. It is intended that this research not only supports computer science educators in their practice, but also models an approach to relating the cognitive phases of learning with educational strategies that in turn may be adapted to other learning domains.

## 1. Introduction

The human brain is a truly amazing object. The approximately three pound mass comprises around 10 billion neurons that regulate cognitive activity. These neurons are in turn surrounded by approximately 100 billion glial cells to provide support, insulation and nourishment (Sylwester, 1995). The cytoarchitecture of the human brain (or way in which these neurons are interconnected) results in the most sophisticated and flexible learning instrument known to humankind. For instance, human infants are capable of learning any of over 3000 languages without initially being proficient in any of them. The human brain is so developed that it is the only matter in the universe capable of self comprehension (that we know of at least!).

Before the current age of brain imaging technology educators did not have the capacity to study actual cognition and so had to adopt a primarily behaviorist approach to analyzing learning. But with the advent of technologies such as CAT scans and Magnetic Resonance Imaging, neuroscience can be used in conjunction with psychological models of learning to provide educators with a more sophisticated understanding of how learning occurs. As Sylwester (1995) comments “can a profession whose charge is defined by the development of an effective and efficient human brain continue to remain uninformed about the brain?” (p. 6).

The purpose of this article is to relate relevant findings from neuroscience and cognitive psychology that will allow Computer Science educators to construct learning resources and activities based on a more scientific understanding of

how learning occurs. It is not the intention of this paper to offer suggestions about how specific computing concepts should be taught – this would be different for each skill and context, and is an enormous task that will not be covered here. However, this article does provide a map to strategies from educational literature that address the general thinking stages involved in many (if not all) learning experiences in computing. The belief is that it is important for educators to at some point be presented with an explanation of brain functioning as it relates to learning to form part of their professional skillset, to allow them to design learning experiences from a more informed and scientific foundation (Rumelhart, 1989).

## 2. Cognitive processes in learning computing

Obviously there are literally millions of cognitive events that occur while a student is engaging in a computing learning task. It will never be possible to describe all of these, and at this stage scientists only have a broad understanding of how different components of the brain operate and interrelate. As well, the type of thinking in which students engage will depend not only on the task but also the student’s approach towards that task. Nevertheless, recent developments in cognitive science have resulted in several models of brain functioning to be proposed. These in turn provide insight into some of the mental processes required for students to learn computing. These processes (or stages) are presented below by way of example, and incorporate aspects of attention, selection, comprehension, retrieval, synthesis, memorization, and abstraction.

Imagine this scenario. Suppose you are teaching an introductory computer programming laboratory class and you want your students to learn about the process of iteration by working on the following task:

*“Write a program that uses a ‘for’ loop to calculate the sum of the first 10 square numbers.”*

What do findings from neuroscience and cognitive psychology tell us about what goes on in the mind of the learner after you have asked them to complete such a question from their laboratory worksheet?

To start with, you need to hope that your instructions have captured the students' *attention*. That is to say, you hope that the verbal information contained in your request for students to complete the question from their practical sheet has been registered by the approximately 30,000 auditory receptors that a typical student possesses (Naarendorp, 2006).

You also hope that student's vigilance network was adequately active to signal their orienting network, which can in turn engage their executive network so that the task description can be processed in working memory (Byrnes, 2001). The right parietal and right frontal lobes are required to maintain a vigilant state (see figure 1). The parietal lobes, the superior colliculus and thalamus are required to shift orientation to a particular stimulus, while the anterior cingulate gyrus is thought to be the key brain structure involved in the executive network (Byrnes, 2001).

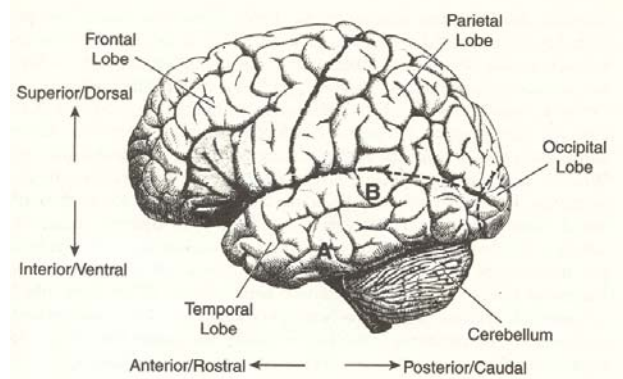


Figure 1. The Cortex Lobes (Byrnes, 2001)

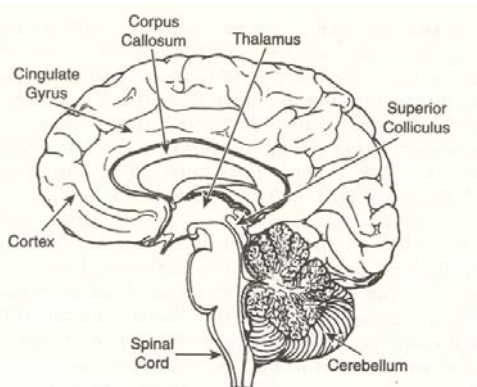


Figure 2. Sub cortical regions (Byrnes, 2001)

Assuming the task description does enter the student's working memory, neuroscientific research indicates that this auditory information has about 2-3 seconds to be interpreted and relevant aspects *selected* for further processing before it

is lost (Anderson, 1995). This involves coordination of the right brain to synthesize the vast amounts of information being received by the students' sensory fields and their left brain to select individual elements from their perceived environment that are identified as important for further analysis (Sylwester, 1995). Experiments suggest that the working memory network includes the prefrontal cortex (in the frontal lobes) and several areas to which this cortex is connected, especially the parietal cortex, hippocampus, and thalamus (Byrnes, 2001). Other studies suggest that tasks involving verbal working memory excite Brodmann Area 40 of the left parietal lobe and three specific Brodmann areas in the left frontal lobe (Byrnes, 2001). An intricate and complex process considering we are still only the stage of selecting relevant information from the environment!

Before students can engage with the learning task, they must first *comprehend* the information contained in the instructions. Once students read the task from their practical sheet, it will need to be correctly interpreted by the orthographic processor (which processes written letters and words), the meaning processor (which interprets the meaning assigned to words) and the context processor (which constructs a coherent interpretation of the task, often called the "referential representation" or "situational model") (Seidenberg & McClelland, 1989). The extra striate area (surrounding the occipital region) is required for orthographic processing, while the Broca area of the frontal lobe and areas in the medial temporal lobe play a pivotal role in semantic processing (Byrnes, 2001).

As well, students need to *retrieve* relevant records (long term memories) relating to the task from their cerebral cortex, which is where permanent memories are stored. Different types of knowledge need to be retrieved from permanent memory:

- *declarative* (or factual) knowledge, such as that relating to the syntax required to write a "for" loop
- *procedural* knowledge, for instance, how to operate the Integrated Development Environment to debug and compile programs
- *conceptual* knowledge, relating to understanding how the mechanics of a "for" loop can be used to design a solution to the task at hand.

This declarative knowledge is primarily stored in the frontal and medial temporal lobes of the cerebral cortex. On the other hand, it is the striatum (anterior and lateral to the thalamus) that is required for storing more procedural skills (Byrnes, 2001).

Not only do these records need to be retrieved and comprehended but they also need to be *synthesized* in an appropriate sequence if a correct solution to the problem is to be derived. To achieve this, the corpus callosum (which facilitates communication between the hemispheres) and the

cingulate gyrus (involved in coordinating several areas of brain functioning) come into play. Only if these regions effectively related all required records in a complete and ordered sequence can the solution be derived.

Once students have constructed the solution, they need to commit explicit aspects of the learning episode to *memory*. Once again structures in the medial temporal lobe are required, including the hippocampus, entorhinal cortex, parahippocampal cortex and perirhinal cortex (Squire & Knowlton, 1995, cited in Byrnes p. 66). The thalamus – the brain’s relay station for cerebral cortex transmissions – is also heavily utilized (see figure 3).

However, for deep learning to have occurred it is not enough for episodic and declarative aspects of the experience to be allocated to memory; translation of the experience into semantic and conceptual forms is required. This *abstraction* requires further activation of the hippocampus (which is located in the medial portion of the temporal lobe) to convert important aspects of the short-term experience into more context free permanent records (Sylwester, 1995). In this way the facts and procedures that underpin the activity can contribute to the learner’s conceptual knowledge base from which future problems can be solved.

This is an exhausting and complex trail of brain activity that identifies just some aspects of brain functioning and only at a broad level. Obviously deconstruction of the thinking required to complete such a task can occur (almost) ad infinitum. For instance, in order to process the mathematic aspects of the problem, the autonomous brain regions that comprehend and produce numeric representations (both in orthographic and verbal form) will be required (McCloskey, Aliminosa, & Sokol, 1991). This involves the use of the inferior occipital-temporal areas and the left perisylvian areas of the brain (Dehaene & Cohen, 1997). Then for actual semantic calculation of the numeric information to occur will require use of the bilateral inferior parietal areas of the brain (Dehaene & Cohen). And so on...

It is not intended that educators remember the specific details of the process described above – it is presented for their information and to provide references if individuals are interested in pursuing particular aspects. The overriding point is this: by understanding the sorts of processes that need to occur in the human mind in order to complete a computing task, educators can design instruction and activities that account for each of these stages. This careful deconstruction of mechanics is important because if students are unable to complete but one of these cerebral processes required for a completing a prescribed learning activity, then unfortunately they will not succeed in understanding.

### 3. So what can be done to assist learning?

Learning has many definitions at many different levels (computational, algorithmic, implementation). At an “implementation” level learning can be considered the forming of permanent records of declarative, procedural and conceptual knowledge, by establishing relatively permanent synaptic connections among relevant neurons (Byrnes, 2001). Using this implementation level definition, capacity to learn computer programming depends on a person’s “experience-dependent plasticity” (Greenough, Black, & Wallace, 1987) which is a neuroscientific term that describes the creation and or reorganization of new synapses.

Educators, however, generally operate at the “computational” and “algorithmic” levels when it comes to learning. They consider what has to be learnt and postulate the steps required to accomplish this, without regard for the underlying cognitive restructuring that needs to occur. This is, of course, perfectly valid – it is not feasible to design learning experiences targeted at specific cytoarchitectural manipulation. However, just as Piaget progressed the field of education by considering the underlying constructs of thinking rather than just the observed inputs and outputs (as was the approach taken by the behaviorist model of education), considering the neuroscientific aspects of cognition provides educators with the opportunity to design educational experiences based on a better understanding of how learning actually occurs.

So the questions becomes, based upon the components of thinking identified by neuroscientific research that are required to complete computer programming tasks, what is it that Computer Science educators can do to best promote learning effective?

The proceeding discussion presents tactics to assist each of the following cognitive stages required for learning:

1. Capturing attention
2. Focusing/selecting important elements from the perceptive field
3. Comprehension
4. Retrieval of relevant records (declarative, procedural, conceptual)
5. Synthesizing and sequencing records (problem solving)
6. Memorizing important aspects of the learning episode
7. Abstracting concepts.

#### 3.1 Attention

Attention is the first stage of Gagne’s (1985) famous Conditions of Learning Model. No learning can occur unless the instructor or task has captured the students’ attention. Vygotsky (1978) proposed two kinds of attention,

“natural attention” and “higher order attention.” While natural attention was said to be largely involuntary and closely linked to immediate perception, higher order attention was proposed to be voluntary, symbolic and strategic (Byrnes, 2001). This higher order or “selective” attention is required for effective learning and can be distinguished by dimensions of orienting, filtering, searching and expecting (Plude, Enns, & Brodeur, 1994).

Given that capturing and maintaining attention is essential for student learning, what strategies can pedagogues execute to facilitate this process? Educational literature offers several recommendations:

1. Cater to learning preferences
2. Appeal to student interests
3. Adopting active processing approaches to instruction
4. Well prepared delivery

There are several different models for classifying learning preferences. For instance preferred sensory perception (Visual, Audio, Kinesthetic, ref: Willingham, 2005), the Felder-Silverman Index of Learning Styles (Sensing versus Intuitive, Visual versus Verbal, Active versus Reflective, Sequential versus Global, ref: Thomas, Ratcliffe, Woodbury, & Jarman, 2002), field dependency (Field Dependent versus Field Independent, ref: Witkin, Moore, Goodenough, & Cox, 1977), and Kolb’s Learning Style Inventory (Activists, Reflectors, Theorists, Pragmatists, ref: Howard, Carver, & Lane, 1996) are all ways of analyzing students’ learning styles. The problem with these models is that in any classroom of students a variety of learning styles will be represented, and in most cases it is not practical to prepare different resources for each type. How these theories are helpful in practice is to remind educators of the various dimensions of learning preferences, so that they may develop instruction and activities catering to all (either simultaneously through multi-faceted delivery or at one time or another in the learning experience).

The importance of appealing to student interest is pervasive across educational literature. Shuell (1986) in his Review of Educational Research article, “Cognitive conceptions of learning,” states:

If students are to learn desired outcomes in a reasonably effective manner then the teacher’s fundamental task is to get students to engage in learning activities that are likely to result in their achieving those outcomes. (p. 429)

Relevance forms a key component of several renown models of teaching including Brown, Collins, and Duguid (1989) – “Situated Cognition” and Knowles (1984) – “Andragogic Theory of Adult Learning.” People have a natural disposition to certain objects, subjects, and themes.

Providing learning experiences based on contemporary topics that are practically relevant to students is crucial to gain and maintain their attention.

Educational researchers such as Wittrock (1991) argue that active processing promotes greater retention because it fosters enhanced attention to the to-be-learned material. The following quote by Glasser (1990) is well publicized for a reason:

... we learn:  
10% of what we read  
20% of what we hear  
30% of what we see  
50% of what we both see and hear  
70% of what we discuss  
80% of what we experience  
95% of what we teach someone else.

It arguably encapsulates the fundamental principle of education - the more actively involved a student, the deeper the learning experience. When students are so engrossed with the learning activity that they lose all sense of time and place, the chance of interference from extraneous sources is minimized and more a more effective learning experience results (Csikszentmihalyi, 1991). Providing student-centred tasks leading to active engagement is another strategy for gaining and maintaining student attention.

Most educators have observed wavering student attention when lesson materials provided are inadequate or the lecturer is not confident with the work. Research by Smith (1985) involving 448 students found that teacher uncertainty was significantly detrimental to student achievement, while lesson discontinuity negatively affected student evaluations of the lesson. Having well prepared instruction and activities increases the likelihood that student attention will be captured and maintained.

As well as all the points made above, it is important for educators to understand the limitations of students’ attention. Research shows that people who are in early stages of learning a skill find it difficult to attend to other aspects of a situation due to the capacity limitations of working memory (Anderson, 1990). It is for this reason that novice programmers would not usually be able to write an unfamiliar programming structure (say, a “for” loop) and listen to other instructions at the same time. Over time, however, the execution of such a skill becomes increasingly automatic and learners would be able to complete such a task without involving as much cognitive load. In this way, working memory capacity is said to be “freed up” by the automatization of skills (Byrnes, 2001). This highlights the importance of separating instructional and practice phases of learning when students are first learning computing skills.

### 3.2 Selection

Any instructional episode or task description contains some information that is relevant to learning, and other information that is superfluous. As a matter of automaticity students will choose to ignore some attributes of the experience being presented to them (such as the accent of the speaker, the background color of the page, the room that they are in, and so on) and select other attributes as important to process (keywords in the task description, hints offered by the instructor). Student progress is hindered when they either ignore information that is important to the task at hand or focus upon information that is irrelevant.

While students need to develop the ability to select relevant information from a problem statement without assistance, facilitators can accelerate pupils' progress by deliberately aiding the selection process. There are several tactics for accomplishing this:

1. Implicit or explicit emphasis
2. Triggering students' conscious selection network
3. Metacognitive reflection upon the selection process

Implicitly emphasizing important aspects of the problem statement or learning activity that require greater attention improves learning (Mayer, 2005b). This sort of "signaling" (or "highlighting") may be static or dynamic. Static signaling takes the form of using visual effects to emphasize important elements of the learning materials, for instance, using "metacommunicative" strategies such as text highlighting, central placement of diagrams, or increasing the size of a learning object (Hatcher, 2003). Dynamic signaling includes using tone of voice or pointer tools to direct student attention to the material upon which they should focus. As well, focusing students' attention upon important elements of the task can also take an explicit form through either textual or verbal references, such as "the aspects of this task that you should be paying careful attention to are..."

Lecturers can also execute strategies to activating students' conscious selection network. Prescribing tasks such as identifying pertinent aspects of a problem statement or prompting students with questions like "what do you perceive to be the key elements of the following program..." can develop their ability to select information relevant to learning. By improving students' selection capacities the level of extraneous cognitive load they will experience is reduced, which increases the amount of working memory they can dedicate to the problem at hand (van Merriënboer & Ayres, 2005).

Metacognition is defined by Slavin (1994) as "monitoring one's own learning behaviors to determine the degree of process and strategies needed for accomplishing

instructional goals" (p. 232). Throughout the instructional process, explicitly educating students about how learning occurs and drawing their attention to the various stages of their own thinking can provide them with skills to intercept ineffective cognitive processes and replace them with efficient strategies. For instance, educators can develop students' metacognitive awareness of selection as a component of problem solving by using asking reflective questions such as "whenever you are given a problem to solve, are how much time do you spend selecting key information and discarding that which is irrelevant? What strategies do you employ to accomplish this?" Such questions help to develop students' control facilities, which in turn effect students' ability to progress to higher levels of thinking (Schoenfeld, 1985).

These are obvious and simple tactics, but their use can sometimes mean the difference between student progressing to comprehend a task as opposed to misunderstand content and requirements. Which specific tactics are employed will depend on the facilitator's assessment of the level of support required by students.

### 3.3 Comprehension

Just because the necessary and sufficient aspects of instruction have filtered through to student's working memory, does not mean that they have achieved comprehension. There are several strategies proposed amongst the literature for assisting comprehension. The following will be briefly discussed:

1. Prescribing tasks of appropriate scope and level
2. Appropriate use of media
3. Providing scaffolding (heuristics, hints, examples)
4. Collaborative learning approaches

In education the "scope" of a learning episode refers to the amount of novel conceptual material introduced (Reigeluth, 1980). An important aspect of instructional design is to determine a scope that will not surpass the information processing capabilities of the target audience but that is also expansive enough to maintain student interest. The "pitch" of a task refers to the level of difficulty of the content being addressed. Materials should be pitched within the students' "Zone of Proximal Development" (Vygotsky, 1978) which is the difficulty level at which students can develop an understanding of concepts based on the scaffolding provided. Providing tasks of appropriate scope and pitch is a prerequisite to effective learning.

Salomon's (1994) Symbol System Theory points out that appropriately matching the medium to the message reduces the level of elaboration and recoding required for learner comprehension. If students require instruction on how to compile a simple "for" loop program, then a page of textual

description or verbal instruction is a far less effective means of delivery than an audio-annotated desktop recording or lecturer demonstration via a data projector. Similarly, experiments in multimedia have indicated that by carefully synthesizing auditory and visual information in a complimentary manner improves comprehension beyond that which can be achieved using one modality alone (Mayer, 2005a).

“Scaffolding” (Vygotsky, 1978) is the provision of resources to assist learners in completing a task that they would not have been able to perform unaided. Scaffolding can take the form of visual cues, verbal definitions, navigational support, textual descriptions, and the like, all aimed at developing students from a state of incomplete to complete understanding. Research validates the use of scaffolding in learning to program. Hendrix, et al. (2000) found that providing students with control structure diagrams to support the reading of computer code lead to increased comprehension. Applin’s (2001) use of program skeletons to support student learning significantly improved student grades in their final examination. The nature of scaffolding prescribed in computer science education will depend on the material requiring comprehension, and is limited only by the instructional designers’ imagination.

Collaborative learning approaches afford opportunities to improve comprehension (Jonassen, Lee, Yang, & Laffey, 2005). Collaborative approaches allow a student’s mental models to be revealed, compared to other students, and as such weaknesses in their understanding to be identified and remedied (either by self, peer or instructor). The instant feedback provided by collaborative approaches allows such misconceptions to be intercepted at an early stage. There are several computer science education studies that evidence improved understanding through collaborative approaches such as cooperative learning (Beck, Chizhik, & McElroy, 2005), pair programming (McDowell, Werner, Bullock, & Fernald, 2002; Nagappan et al., 2003) and Problem Based Learning (Kay, et al., 2000). If collaborative approaches are well implemented they can develop a general culture of questioning and sharing which can in turn improve student comprehension (Chase & Okie, 2000; Jonassen, et al., 2005).

### 3.4 Retrieval

One of the most difficult aspects of learning to program is not being able to retrieve the information required to solve a particular problem. Retrieval may take the form of recall or recognition, with recall being the higher order task because it requires learners to spontaneously remember an item of information rather than merely confirm its occurrence. Tactics that educators can employ to assist the retrieval process include:

1. Providing explicit cues
2. Stimulant questioning
3. Encouraging the use of external memory aids

Cues are aspects of the environment or a rehearsal system that can cause records to be shifted from being in a state of low activation to being in a state of higher activation, allowing the record to become available to working memory (Byrnes, 2001). Pedagogical approaches such as “advanced organizers<sup>1</sup>” or “pre-quizzes” can serve to activate records that will be of use in a proceeding exercise. Even if students have already mastered the skills being tested the time is not wasted; over-learning can lead to automaticity freeing up working memory for the development of higher order thinking abilities.

Appropriately timed questioning (either rhetorical or direct) is another way facilitators can engage student recall (Slavin, 1994): “Remember in class when we discussed the 3 important features of a ‘for’ loop?” Of course, students should be encouraged to create cues for themselves when forming schema. Facilitators can enquire “Did you give yourself a tactic for remembering the important features of a ‘for’ loop when we covered them in class?” This reminds students to take responsibility for managing their recollective abilities.

When beginning programming, using external memory aids such as syntax summaries and typical examples can allow students to more effectively focus their attention upon higher level problem solving and design tasks (Renkl, 2005). However evidence suggests that it is important to fade this type of scaffolding to avoid student dependency on such aids (Sedig, Klawe, & Westrom, 2001; Teles, 1994). Active approaches to memorization can alleviate such learner dependence (see below).

### 3.5 Synthesizing and Sequencing (Problem Solving)

Once relevant permanent records have been retrieved the challenge becomes one of synthesizing and utilizing that information to solve the problem at hand. Learning problem solving is a complex task as it requires coordination of several components of the mind. Due to the dynamic nature of problem solving, collaboration and modeling techniques have proven successful because they allow frequent interchange of ideas, and insight into expert cognition, respectively. Problem solving requires conceptual

---

<sup>1</sup> “Advanced organisers are problems, tasks, or stories presented at the beginning of a lesson general statements given before instruction to orient students to material they are about to learn and to help them recall related information” (Ausubel, 1960).

knowledge, i.e., the ability to understand the meaning and appropriate use of facts and procedures, or “knowing why” (Byrnes, 2001). It is the ability to combine declarative, procedural, and conceptual knowledge in a strategic, efficient and context sensitive way.

Educational literature provides several recommendations for lecturers to develop students’ problem solving skills:

1. Using problem solving teaching heuristics
2. Expert Modeling (Implicit presentation of reasoning sequences, critical thinking, and controls)
3. Explicit explanation of problem solving techniques
4. Metacognitive tasks
5. Collaborative problem solving exercises

Muller, Haberman and Averbuch (2004) present a set of guidelines for teaching problem solving patterns in Computer Science. This may be applied to finding the solution to a specific programming problem or it may be used as a more abstract, general approach to solving problems. It involves the following components:

- a. Providing a representative example of a type of problem
- b. Providing a definition and description
- c. Specifying a pattern name
- d. Identifying similar patterns and similar problems
- e. Comparing to other types of solutions
- f. Identifying typical uses of the pattern
- g. Highlighting common mistakes and difficulties
- h. Pattern composing (for problems whose solutions may be composed of several problems)
- i. Practicing the modification of pattern related solutions to solve alternative problems.

Muller, Haberman, and Averbuch (2004) point out that the use of every stage is not required in all teaching situations that relate to problem solving. However, taken as a whole, the heuristic provides a thorough checklist for teachers.

Observing experts in action is espoused as one of the most effective techniques for developing programming expertise in novice learners (Collins, Brown, & Holum, 1991). “Modeling” offers Computer Science educators the capacity to impart thought processes, problem solving techniques and a whole range of other underlying skills that are not made explicit or at least not embedded in their context when other methods of teaching are employed. Expert modeling offers students a “Cognitive Apprenticeship” (Collins, et al., 1991) through which critical thinking processes are demonstrated and thus can be subsequently adopted by learners. Modeling also has the added advantage of being able to implicitly impart attitudes towards problem solving, such as persistence and enjoyment.

That is not to say the behavior modeling process need necessarily be devoid of explicit explanation. For instance, Landa’s (1976) Algo-heuristic theory can be deployed specifically to support expert modeling. Algo-heuristic theory deconstructs the conscious and especially unconscious mental processes that underlie expert learning, thinking and performance in any domain. This theory presents a system of techniques for getting inside the mind of expert learners and performers to uncover the underlying processes involved. These are then decomposed into elementary components – mental operations and knowledge units – which can in turn be used to teach algorithmic and/or heuristic based tasks. Thus by combining instructor modeling with Algo-heuristic support, students are exposed to both implicit and explicit expertise development mechanisms.

Problem solving is another aspect of cognition particularly suited to metacognitive reflection because of its multifaceted nature and the need to choose an appropriate course of action from a number of possibilities. Ginat (2001) proposes the use of “self-explanation” tasks for the development of metacognition, which forces students to reflect upon and justify their control decisions. In this way learners can critically evaluate the success of their and other students’ problem solving tactics, thus improving their overall problem solving ability on future tasks. Landa (1976) also suggests that tasks requiring students to compose algorithms and heuristics for solving problems requires them to identify the general thinking processes involved in deriving solutions, which promotes metacognitive development.

Finally, collaborative learning allows students to compare and contrast their problem solving techniques to those of others. It affords opportunities for students to learn “vicariously” (Bandura, 1977) through both example and non-example. Such approaches have been met with reported success. For instance, Kay, et al. (2000) found that by adopting a Problem Based Learning approach to teaching introductory programming which involved the pervasive use of collaboration, the mean examination mark improved from 63% to 91% over a two year period. Beck, Chizhik, & McElroy (2005) report significant gains in computing understanding through cooperative learning tasks as opposed to independent learning approaches ( $p = 0.010$ ).

### 3.6 Committing to Memory

One of the most common frustrations that educators recount is teaching a lesson requiring previously learnt knowledge and finding that students have not remembered the ideas and concepts they need. Learning requires encoding, that is, taking sensory information and transforming it into permanent records (Anderson, 1995). But how can educators best ensure that students form a mental

representation of the experiences which are presented to them? There are several techniques suggested by educational literature to assist recollection of facts:

1. Rehearsal
2. Organization
3. Elaboration
4. Consolidation
5. Associative memorization techniques
6. Developing metamemory skills
7. Impactful instruction

Rehearsal is the process of repeating or “re-experiencing” a particular stimulation or thought pattern (Anderson, 1995). Practice affects the “string” of a memory, which is the ease with which it can be retrieved from memory and made available to consciousness. Practice allows semantic declarative memories to become procedural long term memories that are automatic (Sylwester, 1995). Many studies have shown that the amount of practice performed increases the string of a record which in turn results in improved student recollective performance (Byrnes, 2001). This does not mean that students should be continually dragged through meaningless drill exercises – however for skills that require automaticity (such as declarative syntactic knowledge) time spent wrote learning concepts can improve the efficiency with which students tend to higher level tasks.

Organization is “the process of arranging to-be-remembered material into subgroups and hierarchies of subgroups” (Byrnes, 2001, p. 61). This sort of task forces students to analyze where different chunks of knowledge reside in relation to existing concepts. The advantage of organization tasks such as creating concept maps or constructing a “micropedia” of computing terms is that they promote the development and reinforcement of deeper understanding by requiring learners to identify the relationships between components of knowledge.

Elaboration “pertains to the process of ‘going beyond the information given’ and embellishing a raw experience with additional details” (Byrnes, 2001, p. 56). Setting tasks that require a student to elaborate, such as “Write an explanation of how a ‘for’ loop works in conversational language for someone who has never learnt about programming” requires that students reformulate their knowledge, assimilating it into their existing schemata. Such tasks require a deeper level of processing which leads to better retention of material (Anderson, 1995).

Consolidation refers to the revisiting of concepts in order to maintain their place in permanent memory. Findings suggest that memories decay in an exponential manner unless they are revisited (Byrnes, 2001). By simply spacing learning out over the duration of a course concepts are more

easily maintained and students retain the prerequisite knowledge they require for formation of new concepts, meaning less time is lost to relearning. Dempster (1998) describes spacing as “one of the most dependable and replicable phenomena in experimental psychology...with considerable potential for improving classroom learning” (p. 627). By attempting a similar task two days later and then again two weeks later strong memories can be formed that may have otherwise being forgotten due to decay or loss of retrieval cues.

Taken together, these first four techniques for committing to memory (rehearsal, organization, elaboration, and consolidation) represent the dimensions of the Levels of Processing Theory, which proposes that the more attention a concept receives, the greater its strength in memory. It should be noted, however, that there are other tangential approaches for committing information to memory, based upon associative techniques. Pegword pneumonics, the “keyword method” and the “method of loci” are three popular approaches for facilitating memorization and recall (Slavin, 1994). By applying such techniques to computing curriculum students not only learn the declarative knowledge or heuristics being addressed but also about how to apply memory strategies to their learning.

One important type of metacognition is “metamemory.” Metamemory refers to a person’s knowledge and beliefs about how their memory works (Flavell, Miller, & Miller, 1993). Students may be challenged with “Of the strategies identified above, which are the most appropriate for learning syntax?” and “what sort of techniques would be most effective for you when studying for an exam?” Students who are aware of their memory’s functioning and have developed tactics for performing memorization tasks have an obvious advantage over those who do not.

Designing impactful instruction that burns concepts into the minds of the learner will assist the retrieval of the concepts at later stages. Metaphor, analogy, diagrams, catering to different learning styles, and providing multiple representations of information (Dual Code Theory) are all techniques that enhancing encoding and retention. Emotionally charged learning experiences are much more likely to be recalled, referred to as the “flashbulb” memory phenomenon (Byrnes, 2001). It is also worth pointing out that designing instruction similar in form to the conditions under which the material will be called for has been shown to improve the likelihood of retention (Bransford, et al., 1982).

It is important to note that different students will have different innate levels of ability at remembering different types of knowledge. Students with natural declarative memorization abilities will more quickly acquire

lexical/syntactic knowledge of programming languages. Those with strong procedural aptitude will more quickly become familiar with semantic attributes of a language and the tasks associated with programming (like operation of the IDE). Students with high level conceptual abilities will more readily excel at problem-solving and design tasks. Identifying these different components of thinking associated with learning to program allows instructors to better diagnose and remediate student difficulties, and helps pupils to identify and address the area of learning that causes them the most concern.

Another point worth reflecting upon is the extent to which effort should be applied committing different types of computing skills to memory. For instance, through frequent and deliberate revisiting of linguistic aspects of programming (syntax) students can develop a level of automaticity that will allow them to retrieve constructs more easily. This makes sense, as it frees up students' working memory to address higher level ideas and concepts. Some effort may be spent committing common algorithmic tasks to memory, offering students a template from which to construct solutions for semantically similar situations. However because of the degree of variability between algorithmic implementations automaticity needs to be balanced with understanding. Finally, it would be unwise to spend time committing design solutions to memory for the purposes of developing student design skills – tasks involving creativity and reflectivity require understanding rather than memorization due to their strong dependence on context.

### 3.7 Abstraction

Hazzan (2003) presents three definitions of abstraction:

1. abstraction level as the quality of the relationships between the object of thought and the thinking person (Wilensky, 1991);
2. abstraction level as reflection of the process-object duality (Dubinsky, 1991; Sfard, 1991);
3. abstraction level as the degree of complexity of the concept of thought. (p. 97)

In other words, abstraction is the process of generalizing specific concrete memories to applications other than the one experienced in the learning context. Greater abstraction increases the usefulness of learning experiences by increasing the range of situations within which concepts can be utilized. Kurland, et al. (1989) propose that the very reason novices experience difficulty transferring concepts across contexts is because of the way they classify similarities and differences in various applications and structures. It is because novices classify tasks and according to more superficial and concrete characteristics such as

language and context rather than underlying conceptual structures the locus of their transfer is reduced.

The cyclic Actions-Process-Object model is an interdisciplinary explanation that cognitive psychologists use to describe how abstraction occurs. Under this model people abstract by transforming processes into objects.

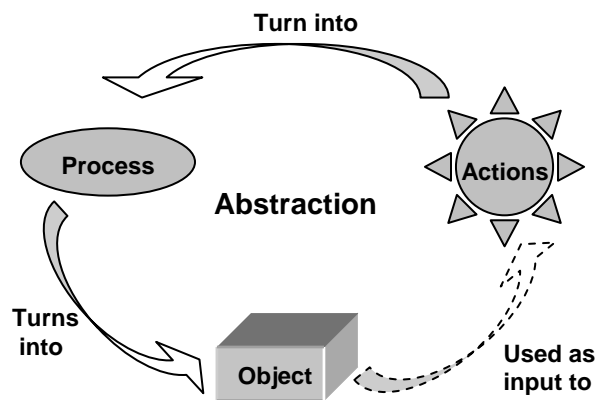


Figure 3. A simplified version of the Actions-Process-Object model (Ahanori, 2000, p. 28)

Using this framework, facilitating abstraction becomes a task of providing activities and learning experiences that allow people to utilize and reflect upon sets of actions that form processes, which can then in turn be considered as objects for future actions. For instance, asking students to construct several “for” loops on a worksheet allows them to become familiar with the process, which can then form a “for” loop object for use in writing future programs.

Literature provides several tactics for developing student abstraction:

1. Constructivism
2. Chunking
3. Reflective tasks requiring the development of more complex mental representations
4. Proposing abstraction barriers to facilitate incremental development of schema
5. Multiple, context dependent case studies

Constructivist approaches to learning (Piaget, Vygotsky) propose that “learners must individually discover and transform complex information, checking new information against old rules and revising them when they no longer work” (Slavin, 1994). A “problem-centred” rather than “answer-centred” approach allows learners to develop a “relational” understanding (knowing both what to do and why) rather than merely an “instrumental” understanding (rules without reasons) (Skemp, 1976). The logico-deductive nature of computer programming requires that students have an understanding of how concepts interrelate

in order to develop accurate mental representations. Carefully designing curriculum on a constructivist basis allows these interrelations to be understood so that new concepts can be assimilated into existing schema – a prerequisite for abstraction to occur.

Students can overcome the limits of cognitive capacity when dealing with complex programs by using “chunking” (Rist, 1989). As small pieces of knowledge are combine into lines of code, lines of code into simple plans, simple plans into more complex plans and programs, the chunks of knowledge constructed at each level of description hide the complexity of the level below it. As students develop expertise the task of writing programs becomes simpler because chunks of knowledge can be retrieved rather than created from scratch (Rist, 1989). Instructors can take an explicit approach to chunking by pointing out how a particular approach forms a design pattern in computing, and providing examples of how it may be applied. Lecturers may choose to set comprehension tasks that require students to read code and form chunks, and point out to students the value of consciously adopting this as a private study technique.

Hazzan (2003) draws upon the Action-Process-Object model when she proposes the importance of “reflective abstraction” tasks for ensuring students develop complete representations of abstract concepts. In her research she found that the quality of relationship between the object of thought and the person is compromised as students “reduce abstraction,” which is the unconscious process of finding ways to make an unfamiliar idea more familiar by applying the concept to objects that they already know. Thus instructors need to ensure that they include tasks that encourage students to consciously reflect upon the mental models in an expansive and elaborative manner. By designing activities that oblige students to articulate their models completely the complexity and sufficiency of their representations can be evaluated by teachers, peers, and themselves.

Ahanori (2000) augments the Action-Process-Object model for computer science thinking to suggest the posing “abstraction barriers” as an effective way to develop Program-Free Thinking abilities. For instance, hiding data structure implementations from students so that they can experiment with the data structure operations provides a barrier that allows students to form abstract cognitive schema about the functioning and use of data structures without having to engage in inordinate amounts Programming-Language Oriented Thinking. In this way cognitive load is reduced, allowing a focus on abstraction without being burdened by aspects of implementation.

Spiro, Feltovitch, and Coulsons’ (1988) Cognitive Flexibility Theory suggests using multiple context-dependent case studies that reflect the complexity faced by industry practitioners (rather than presenting cases requiring simple linear decision making processes) as a way to develop the pervasiveness of student’s mental models. Multiple context dependent case studies emphasize inter-connectedness and avoid oversimplification of abstractions, allowing students to develop an appreciation of the intricacies of complex domains as opposed to trying to compartmentalize the subject matter. It also has the advantage of counteracting the “reduce abstraction” tendency (which Feltovich, 1997, refers to as the “reductive bias”) by forcing students to elaborate their schema. Armed with more sophisticated abstractions students can more easily restructure their knowledge in response to changing situational demands.

#### **4. Conclusion**

The brilliant sophistication of the human mind inevitably implies its sensitivity – a lot needs to go right for effective learning to occur. Neuroscience has contributed a great deal to psychological models of learning by providing justification (or refutation) of the processes underpinning learning. Teachers generally and computer science educators specifically can capitalize on these findings to provide learning experiences that cater to all the identified cognitive stages of learning, thus enhancing the effectiveness of their instruction.

By understanding the components of learning that cognitive research has uncovered, lecturers are more equipped to approach learning design from a scientific and informed point of view. While at this stage neuroscience may not provide all the answers on how to best educate our computing students, the process of reflecting upon the models of thinking that are being proposed offers teachers a valuable catalyst for refining their practices. It is intended that by articulating the cognitive stages that underpin learning and mapping strategies from educational literature to each of these, computer science educators may more effectively practice their profession.

#### **References**

- Aharoni, D. (2000). Cogito, Ergo sum! cognitive processes of students dealing with data structures. *Proceedings of the thirty-first SIGCSE technical symposium on Computer science education*, 26-30.
- Anderson, J. R. (1990). *Cognitive psychology and its implications* (3rd ed.). New York: Freeman.
- Anderson, J. R. (1995). *Learning and memory: An integrated approach*. New York: Wiley.
- Applin, A. G. (2001). Second language acquisition and CS1. Paper presented at the Proceedings of the thirty-

- second SIGCSE technical symposium on Computer Science Education, Charlotte, North Carolina, United States.
- Ausubel, D. P. (1960). The use of advanced organisers in the learning and retention of meaningful verbal material. *Journal of Educational Psychology*, 51, 267-272.
- Bandura, A. (1977). *Social learning theory*. New York: General Learning Press.
- Beck, L. L., Chizhik, A. W., & McElroy, A. C. (2005). Cooperative learning techniques in CS1: Design and experimental evaluation. Paper presented at the Proceedings of the 36th SIGCSE technical symposium on Computer science education, St. Louis, Missouri, USA.
- Bransford, J. D., Stein, B. S., Vye, N. J., Franks, J. J., Auble, P. M., Mezynski, K. J., & Perfetto, G. A. (1982). Differences in approaches to learning: An overview. *Journal of Experimental Psychology: General*, 3, 390-398.
- Brown, J. S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. *Educational Researcher*, 18(1), 32-42.
- Byrnes, J. P. (2001). *Minds, brains and learning*. New York: The Guilford Press.
- Chase, J. D., & Okie, E. G. (2000). Combining cooperative learning and peer instruction in introductory computer science. Paper presented at the Proceedings of the thirty-first SIGCSE technical symposium on Computer science education, Austin, Texas, United States.
- Collins, A., Brown, J., & Holum, A. (1991). Cognitive apprenticeship: Making thinking visible. *American Educator*, 6(11), 38-46.
- Csikszentmihalyi, M. (1991). *Flow: The psychology of optimal experience*. New York: Harper Perennial.
- Dehaene, S., & Cohen, L. (1997). Cerebral Pathways for calculation: Double dissociation between rote verbal and quantitative knowledge of arithmetic. *Cortex*, 33, 219-250.
- Dempster, F. (1998). The spacing effect: A case study in the failure to apply the results of psychological research. *American Psychologist*, 43, 627-634.
- Flavell, J. H., Miller, P. H., & Miller, S. A. (1993). *Cognitive development*. Englewood Cliffs, NJ: Prentice Hall.
- Gagné, R. (1985). *The conditions of learning* (4th ed.). New York: Holt, Rinehart & Winston.
- Ginat, D. (2001). Metacognitive awareness utilized for learning control elements in algorithmic problem solving. *SIGCSE Bull.*, 33(3), 81-84.
- Greenough, W. T., Black, J. E., & Wallace, C. S. (1987). Experience and Brain Development. *Child Development*, 58(3), 539-559.
- Hatcher, S. (2003). Reading between the lines: Metacommunicative aspects of online education. Paper presented at the 111th Annual Conference of the American Psychological Association, Toronto, Canada.
- Hazzan, O. (2003). How students attempt to reduce abstraction in the learning of mathematics and in the learning of computer science. *Computer Science Education*, 13(2), 95-123.
- Hendrix, T. D., II, J. H. C., Maghsoodloo, S., & McKinney, M. L. (2000). Do visualizations improve program comprehensibility? Experiments with control structure diagrams for Java. *Proceedings of the thirty-first SIGCSE technical symposium on Computer science education* (pp. 382-386): ACM Press.
- Howard, R. A., Carver, C. A., & Lane, W. D. (1996). Felder's learning styles, Bloom's taxonomy, and the Kolb learning cycle: Tying it all together in the CS2 course. Paper presented at the Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education, Philadelphia, Pennsylvania, United States.
- Jonassen, D. H., Lee, C. B., Yang, C.-C., & Laffey, J. (2005). The collaboration principle in multimedia learning. In R. E. Mayer (Ed.), *The Cambridge Handbook of Multimedia Learning* (pp. 247-270). New York: Cambridge University Press.
- Kay, J., Barg, M., Fekete, A., Greening, T., Hollands, O., Kingston, J. H., & Crawford, K. (2000). Problem-based learning for foundation computer science courses. *Computer Science Education*, 10(2), 109-128.
- Knowles. (1984). *The adult learner: A neglected species* (3rd ed.). Houston: Gulf Publishing.
- Kurland, D., Pea, R., Clement, C., & Mawby, R. (1989). A study of the development of programming ability and thinking skills in high school students. In E. Soloway & J. C. Spohr (Eds.), *Studying the Novice Programmer* (pp. 83-112). Hillsdale, NJ: Lawrence Erlbaum.
- Landa, L. (1976). *Instructional regulation and control: Cybernetics, algorithmization, and heuristics in education*. Englewood Cliffs, NJ: Educational Technology Publications.
- Mayer, R. E. (2005a). Introduction to multimedia learning. In R. E. Mayer (Ed.), *The Cambridge Handbook of Multimedia Learning* (pp. 1-17). New York: Cambridge University Press.
- Mayer, R. E. (2005b). Principles for reducing extraneous processing in multimedia learning: Coherence, signaling, redundancy, spatial contiguity, and temporal contiguity principles. In R. E. Mayer (Ed.), *The Cambridge Handbook of Multimedia Learning* (pp. 1-17). New York: Cambridge University Press.
- McCloskey, M., Alimoso, D., & Sokol, S. M. (1991). Facts, rules, and procedures in normal calculation: Evidence from multiple single patient studies of impaired arithmetic fact retrieval. *Brain and Cognition*, 17, 154-203.
- McDowell, C., Werner, L., Bullock, H., & Fernald, J. (2002). The effects of pair-programming on performance in an introductory programming course, *Proceedings of*

- the 33rd SIGCSE technical symposium on Computer science education (pp. 38-42): ACM Press.
- Muller, O., Haberman, B., & Averbuch, H. (2004). (An almost) pedagogical pattern for pattern-based problem-solving instruction. *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education* (pp. 102-106): ACM Press.
- Naarendorp, F. (2006). Northeastern University introduction to sensation and perception - hearing. Retrieved September 1, 2006, from <http://www.psych.neu.edu/Courses/Syllabi/PSYU452/hearing.html>
- Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., & Balik, S. (2003). Improving the CS1 experience with pair programming. Paper presented at the Proceedings of the 34th SIGCSE technical symposium on computer science education.
- Plude, D. J., Enns, J. T., & Brodeur, D. (1994). The development of selective attention: A life-span view. *Acta Psychologica*, 86, 227-272.
- Reigeluth, C. M. (1980). The elaboration theory of Instruction: A model for sequencing and synthesizing instruction. *Instructional Science*, 9(3), 195-219.
- Renkl, A. (2005). The worked-out examples principle in multimedia learning. In R. E. Mayer (Ed.), *The Cambridge Handbook of Multimedia Learning* (pp. 249-245). New York: Cambridge University Press.
- Rist, R. S. (1989). Schema creation in programming. *Cognitive Science*, 13, 389-414.
- Rumelhart, D. E. (1989). The architecture of the mind: A connectionist approach. In M. I. Posner (Ed.), *Foundations of Cognitive Science* (pp. 133-160). Cambridge, MA: MIT Press.
- Salomon, G. (1994). *Interaction of media, cognition, and learning*. New Jersey: LEA.
- Schoenfeld, A. (1985). *Mathematical Problem solving*. New York: Academic Press.
- Sedig, K., Klawe, M., & Westrom, M. (2001). Role of interface manipulation style and scaffolding on cognition and concept learning in learnware. *ACM Trans. Comput.-Hum. Interact.*, 8(1), 34-59.
- Seidenberg, M. S., & McClelland, J. L. (1989). A distributed, developmental model of word recognition and naming. *Psychological Review*, 96, 523-568.
- Shuell, T. J. (1986). Cognitive conceptions of learning. *Review of Educational Research*, 56(4), 411-436.
- Skemp, R. (1976). Relational understanding and instrumental understanding. *Mathematics Teaching*, 77, 20-26.
- Slavin, R. E. (1994). *Educational psychology* (5th Ed. ed.). Boston: Allyn and Bacon.
- Smith, L. R. (1985). Teacher clarifying behaviours: Effects on student achievement and perceptions. *Journal of Experimental Education*, 53(3), 162-169.
- Spiro, R. J., Coulson, R. L., Feltovich, P. J., & Anderson, D. (1988). Cognitive flexibility theory: Advanced knowledge acquisition in ill-structured domains. Paper presented at the Proceedings of the 10th Annual Conference of the Cognitive Science Society.
- Sylwester, R. (1995). *A celebration of neurons*. Alexandria: Association for Supervision and Curriculum Development.
- Teles, L. (1994). Cognitive apprenticeship on global networks. In L. Harasim (Ed.), *Global networks: Computers and International Communications* (pp. 271-282): Cambridge, Massachusetts: The MIT Press.
- Thomas, L., Ratcliffe, M., Woodbury, J., & Jarman, E. (2002). Learning styles and performance in the introductory programming sequence. Paper presented at the Proceedings of the 33rd SIGCSE technical symposium on computer science education.
- van Merriënboer, J. J. G., & Ayres, P. (2005). Research on cognitive load theory and its design implications for e-learning. *Educational Technology Research & Development*, 53(3), 5-13.
- Vygotsky, L. S. (1978). *Mind in society*: Cambridge, MA: Harvard University Press.
- Willingham, D. T. (2005). Do Visual, Auditory, and Kinesthetic Learners Need Visual, Auditory, and Kinesthetic Instruction? Retrieved October 2006, from [http://www.aft.org/pubs-reports/american\\_educator/issues/summer2005/cogsci.htm](http://www.aft.org/pubs-reports/american_educator/issues/summer2005/cogsci.htm)
- Witkin, H. A., Moore, C. A., Goodenough, D. R., & Cox, P. W. (1977). Field-dependent and field-independent cognitive styles and their educational implications. *Review of Educational Research*, 47, 1-64.
- Wittrock, M. C. (1991). Relationships among educational research and neural and cognitive sciences. *Learning and Individual Differences*, 3, 257-263.

## Author Information

Matt Bower

Computing Department  
Macquarie University  
Sydney, NSW, 2109  
Australia

[mbower@ics.mq.edu.au](mailto:mbower@ics.mq.edu.au)

<http://www.ics.mq.edu.au/~mbower>

Matt deliberately positions his work between the Computer Science Education and Technology Based Learning fields. His PhD thesis relates to how Web conferencing environments can best be used to facilitate collaborative learning in computer science classes. In his spare time Matt enjoys surfing, skiing, ocean swimming, and pursues an interest in Buddhism. Matt is a Capricorn.