Published in *Instructional Science*, **23**, 133–161 (1995).

Cognitive apprenticeship and its application to the teaching of Smalltalk in a multimedia interactive learning environment

YAM SAN CHEE
Department of Information Systems and Computer Science, National University of Singapore, Lower Kent Ridge Road, Singapore 0511

Abstract. With the rise of the situated cognition paradigm in cognitive science, cognitive apprenticeship has become increasingly prominent as a model of instruction. This paper critically reviews traditional approaches to learning and education to motivate the need for fresh ways of thinking about these topics. Cognitive apprenticeship as an approach to improving learning and education is described. We present an overview of *SMALLTALKER*, a learning environment for Smalltalk programming, and illustrate how the instructional methods of cognitive apprenticeship have been applied in developing that system. The paper concludes with a discussion of issues related to system development effort, evaluation, perceived limitations, and plans for related work.

Introduction

In recent years, cognitive apprenticeship (Brown *et al*, 1988; Burger and DeSoi, 1992; Clancey, 1992b; Collins *et al*, 1989) has become increasingly prominent as a model of instruction. This development is attributable to its potential to help solve the educational problems of brittle skills and inert knowledge that so often arise with traditional schooling (Bransford *et al*, 1989; Resnick, 1987; Whitehead, 1929). Recent research in the learning sciences coupled with a shift to the situated cognition paradigm in the cognitive sciences has led to a significant rethinking of the nature of learning and cognition (Brown, 1990; Brown *et al*, 1988; Clancey and Roschelle, 1991; Lave, 1988; Lave and Wenger, 1991; Rogoff and Lave, 1984) and how we can use technology to support learning (Brown, 1985a; Clancey, 1992a; Collins, 1991; Schank and Edelson, 1989/90). Cognitive apprenticeship is one manifestation of this new way of thinking about learning.

Over the last three to five years, we have also witnessed a surge in the amount of research and the number of publications on object-oriented programming. Object-oriented programming is emerging as the dominant programming paradigm. An important object-oriented programming language is Smalltalk. Smalltalk is a pure object-oriented language with a long history dating back to 1972. It is one of the fastest-growing software languages today and has been described as the *Computer Programming Language of the 90s* (Shafer and Ritz, 1991). With claims of enhanced code reusability and program maintainability, active interest in Smalltalk is not surprising. Both new and existing programmers are anxious to master object-oriented programming technology so as to maintain their productive and competitive edge. However, Smalltalk is not easy to learn (LaLonde and Pugh, 1990a; Nielsen and Richards, 1989). For this reason, we chose to focus our attention on Smalltalk programming.

In this paper, we describe our efforts in applying the principles of cognitive apprenticeship to the design and implementation of *SMALLTALKER*, a Macintosh-based multimedia interactive learning environment for teaching Smalltalk to programming novices. We begin with a critical review of weaknesses in traditional approaches to learning and education. Then, we provide a detailed description of cognitive apprenticeship. The next section gives an overview of the *SMALLTALKER* system, and the section following illustrates how we have attempted to instantiate the principles of cognitive apprenticeship in the system. We then discuss issues related to system development effort, evaluation, perceived limitations, and plans for related work. We conclude the paper with a summary.

Traditional learning and education

The practice of schooling has become a culture unto itself. Resnick (1987) identifies four dimensions along which school learning differs from non-school learning. They are: (a) individual cognition in school versus shared cognition outside, (b) pure mentation in school versus tool manipulation outside, (c) symbol manipulation in school versus contextualized reasoning outside, and (d) generalized learning in school versus situation-specific competencies outside. In school, therefore, emphasis has been placed upon the learning of the individual, with a high premium placed upon activities involving “pure thought” (Resnick, 1987). Much of school learning is symbol-based, removed from the physical world of objects and events that the symbols represent. In an effort to teach general, widely usable skills and theoretical principles, situation-specific learning has been avoided. Growing evidence indicates that such schooling practice results in students being unable to make use of what they learn *in* school *outside* of school. Knowledge remains inert, it is not spontaneously accessed, and its relevance to problem solving situations remains unnoticed (Bransford *et al*, 1989) .

Pea (1992b) identifies the following set of problems: (a) an epistemology that treats students as receivers of knowledge-as-facts, (b) a perspective on learning and teaching as a decontextualized classroom activity, (c) a curriculum-centered view of educational materials and the use of decontextualized tasks for learning basic skills, and (d) a view of teaching as telling or “delivering” curricula. The foregoing perspective on learning and education is referred to as a *transmissional* view because it assumes that knowledge is a *thing* that can be simply transmitted from one person to another and that learning occurs when transmission takes place.

Furthermore, Collins *et al* (1989) argue that existing pedagogical practices place too little emphasis on higher-order problem solving activities that require students to actively integrate and appropriately apply subskills and conceptual knowledge. Consequently, conceptual and problem solving knowledge acquired in school remains unintegrated and inert for many students. Knowledge remains bound to surface features of problems encountered in textbooks and lectures. Key aspects of expertise remain invisible to students because too little attention is paid to the processes that experts engage in to use or acquire knowledge in performing realistic tasks.

Traditional schooling typically fails to support students' metacognitive development. Metacognition can be thought of in terms of cognitive self-appraisal and cognitive self-management (Paris and Winograd, 1990) . Cognitive self-appraisal refers to personal reflections about one's knowledge states and abilitites. An example of cognitive self-appraisal is the question "Can I derive a formula to compute the area of a trapezoid?" Cognitive self-management refers to how metacognition helps to orchestrate cognitive aspects of problem solving; that is, behaviors related to evaluating, planning, and regulating problem solving. The development of metacognition is essential for achieving higher-order, independent thinking. As Paris and Winograd (1990) make clear, academic learning and instruction cannot be separated from the affective and motivational dimensions of learning experience. However, attention to metacognitive development can help to promote students' positive self-perceptions, affect, and motivation. Regrettably, the development of students' metacognitive skills has rarely been an explicit goal of instruction in schools.

The evaluation of learning outcomes has also received severe criticism. Schank and Edelson (1989/90) claim that student instruction and evaluation in schools has been shaped, in part, by the convenience afforded by technology. Multiple-choice tests are widespread. Given that a student need only select one among a set of given answers in a multiple-choice test instead of generating his own answer, such a test is, at best, a limited evaluation tool. Worse still, the multiple-choice test and the completion of workbooks are increasingly used not just as evaluation tools but also as instructional tools. Unfortunately, the use of workbooks makes "all of school look like an exercise in problem solving by the application of known formulas, rather than the encouragement of original thinking and the expression of that thinking" (Schank and Edelson, 1989/90: 5) .

The foregoing criticisms highlight an urgent need to reshape the practice of education. Existing teaching practices emphasize the "correct answer" without reinforcing the structure of the domain being taught. They favour the didactic view of instruction that assumes an objective world of knowledge with "right" and "wrong" answers. They ignore the importance of focusing on the underlying learning *process* rather than the *product* of a creative effort (Brown, 1985b) . In addition, evaluation formats encourage students to memorize answers instead of learning the process that will enable them to generate the answer (Schank and Edelson, 1989/90) . Such practices ill-prepare students for real life where they must generate their own solutions to problems that they must identify themselves.

While the above criticisms are ostensibly targeted at school-based learning, they apply equally to research efforts directed at conventional approaches to integrating computer use into educational practice. The fundamental problem is that, too often, researchers have attempted to *reproduce* existing educational practice in a new medium—the computer. In so doing, they make the mistake of assuming that because a teaching strategy is part of existing practice, it represents sound practice (Schank and Edelson, 1989/90) .

Cognitive apprenticeship

Education needs an approach to teach students to learn *how* to learn (Feinstein, 1988) . Recent progress in learning science supports a constructivist (Chapman, 1988; Forman and Pufall, 1988; Jonassen, 1991; Smith *et al*, 1993) and *transformational* view of knowledge building (Pea, 1992b) . In particular, the transformational view is based on: (a) an epistemology that views knowledge as socially constructed through action, communication, and reflection involving learners, (b) a framework establishing connections between teaching–learning processes and increasing student membership among communities of practitioners outside the traditional classroom, (c) a learner-centered view of educational materials, beginning with tasks that enable instructors to start with what the learner knows and constructing new understanding based on it while working on authentic tasks, and (d) viewing teaching as modeling expert practice and promoting learning conversations that negotiate meanings to promote change in learner concepts and strategies toward proficient performance (Pea, 1992b) .

Schank and Jona (1991) have proposed a classification of six teaching methods, and one of these is the apprenticeship method. (The other methods are referred to as the sponge method, the artist method, the research method, the exploration method, and the argument method.) Traditional apprenticeships focus on specific methods for carrying out domain tasks. Apprentices learn these methods through observation, coaching, and practice. A key aspect of coaching is the provision of scaffolding—in the form of reminders and help—that apprentices receive in their attempt to execute the composite skills demonstrated by masters, coupled with fading—the reduction of support by the master as the apprentice acquires the target skill. A second important aspect of apprenticeship is that learning occurs in an embedded social context. As the embedded context is usually an authentic context for the target skill, transfer of learning becomes unnecessary.

Cognitive apprenticeship, however, has two special emphases (Collins *et al*, 1989) . First, it is directed at teaching processes that experts use to handle complex tasks. When it is necessary to teach conceptual and factual knowledge in pursuance of task accomplishment, cognitive apprenticeship requires that such knowledge be exemplified and situated in the contexts of their use. This approach encourages deeper understanding of the meaning of the concepts and facts; it also develops a rich web of memorable associations between the conceptual and factual knowledge learned and their problem solving contexts. Second, cognitive apprenticeship focuses on the development of cognitive and metacognitive processes in learning rather than the physical skills and processes. This focus requires supporting the externalization of processes that are carried out internally (that is, mentally) and hence are not readily observable. Cognitive apprenticeship teaching methods bring tacit processes out into the open so that they can be observed. In addition, it seeks to encourage the development of self-correcting and self-monitoring skills through the techniques of reflection and engagement in generative–evaluative cycles of learning activity.

The cognitive apprenticeship technique, as formulated by Collins *et al* (1989), consists of six teaching methods: *modeling*, *coaching*, *scaffolding*, *articulation*, *reflection*, and *exploration*. The six methods, in turn, break down into three groups. The first group—modeling, coaching, and scaffolding—represents the core and is designed to help students acquire an integrated set of cognitive and metacognitive skills through observation and supported practice. The second group—articulation and reflection—is designed to focus students' observations of expert problem solving and to gain control of their own problem solving strategies. The final group—exploration—is intended to encourage learner autonomy and problem formulation by the self.

In *modeling*, an expert performs a task so that students can observe his actions and build a conceptual model of the processes required for task accomplishment. The provision of a conceptual model contributes significantly to success in teaching complex skills without resorting to lengthy practice of isolated subskills. In cognitive domains, this necessitates the externalization of internal cognitive processes. Tacit processes are brought into the open so that students can observe, enact, and practise the requisite skills. Modeling techniques have been used particularly effectively in the domains of reading (Brown and Palincsar, 1989) and writing (Scardamalia *et al*, 1984) .

In *coaching*, students are engaged in problem-solving activities that require them to appropriately apply and actively integrate subskills and conceptual knowledge. In this way, conceptual and factual knowledge are exemplified and situated in their contexts of use, thereby grounding the knowledge in experience and making learning meaningful. Consequently, this approach helps to avoid learning outcomes where knowledge remains bound to surface features of problems as they appear in textbooks. The expert coaches students by providing hints, feedback, and reminders to assist students to perform closer to his level of accomplishment.

Coaching is made effective by means of highly interactive and situated feedback. The content of coaching interaction is related to specific problems that students face while carrying out a task.

In *scaffolding,* an expert assists students to manage complex task performance. If necessary, he completes those parts of the task that students have not yet mastered. This method may entail students engaging in legitimate peripheral participation (Lave and Wenger, 1991) ; that is, students participate in the practice of an expert, but only to the extent that they can handle and with the amount of responsibility that they are capable of assuming. Scaffolding is coupled with *fading*, the gradual removal of the expert's support as students learn to manage more of the task on their own. The interplay between observation, scaffolding, and increasingly independent practice aids students in developing the metacognitive skills of self-monitoring and self-correction and in achieving integrated skills and knowledge characteristic of expertise. Thus, modeling and coaching support students' efforts to "grow into" domain competence while scaffolding and fading support students' efforts to "grow out of" dependence on the expert (Scardamalia and Bereiter, 1991) .

In *articulation,* an expert encourages students to explicate their knowledge, reasoning, and problem solving strategies. Such activities provide the impetus for students to engage in the refinement and reorganization of knowledge. The use of synthetic and design tasks in a producer–critic framework is particularly effective in achieving articulation (Allen, 1992; Pea, 1991) . Such tasks require students to participate in generating knowledge and evaluating the outcomes of knowledge-building activities as part of collaborative learning activities. Generative and evaluative processes provide a further basis for concept assimilation and internalization (Lawrence and Valsiner, 1993; Vygotsky, 1978) .

In *reflection,* the expert provokes students to compare their problem solving processes with his own, with that of other students, and with an internal cognitive model of the relevant expertise. Such comparisons aid students in diagnosing their difficulties and in incrementally adjusting their performance until they achieve competence. Reflection is facilitated by the provision of abstracted replay that contrasts students' own performance with that of the expert (Collins and Brown, 1988) .

In *exploration,* the expert pushes students to be independent learners. Students are only set general goals. At the same time, they are encouraged to identify personal interests and pursue personal goals. Forcing students to engage in exploration teaches them how to frame interesting questions and to identify difficult problems on their own.

Cognitive apprenticeship is founded upon the situated cognition paradigm (Brown *et al*, 1988; Clancey, 1991a; Clancey, 1991c; Clancey, 1992b; Clancey and Roschelle, 1991) in cognitive science. This paradigm contrasts markedly with the cognitivistic view of psychology (Still and Costall, 1991) and represents a significant departure from old ways of thinking about cognition (Sandberg and Wielinga, 1992) . (See also *Cognitive Science*, *17* (1), a special issue that addresses the paradigm debate.) Perhaps the most attractive feature of situated cognition rests in its attempt to respect both what is known from neuroscience and sociology (Clancey, 1991b; Roschelle and Clancey, 1992) , thus giving rise to a psychology grounded in physical (neural) and social reality.

Situated cognition places a strong emphasis on how representations are created and given meaning (Clancey and Roschelle, 1991) . There is a bias in favour of addressing the difficult issue of human *understanding* as opposed to simply *knowing.* A serious attempt is made to tackle the problem of semantics instead of sidestepping the problem through the use of knowledge representations (with their imputed semantics). Thus, understanding is viewed in terms of appropriation and meaning negotiation (Pea, 1992a) and entails engaging in sense-making activity (Carroll, 1990; Chee *et al*, 1993) .

Furthermore, Burger and DeSoi (1992) quote Edmondson (1985: 101) on how the philosopher Giambattista Vico of Naples (1668–1744) originated the doctrine which holds that "in order for us really to understand anything, and not merely to know it, it is necessary that we should have made it. Understanding involves creative activity, not necessarily creation of some material artefact, but equally the construction of abstract 'models,' theories, structures in words or in symbols." We see in this, again, the Piagetian concept of constructivism. Harel & Papert (1991) proceed one step further in advocating constructionism. Constructionism shares constructivism's connotation of learning as involving building knowledge structures but adds the idea that knowledge building happens felicitously in a context where learners are consciously engaged in constructing a public entity. The congruence of these ideas with the apprenticeship emphasis on using authentic and synthetic tasks for knowledge building are readily apparent.

The writings of Vygotsky (Vygotsky, 1978; Vygotsky, 1987) also exert a strong influence in shaping the method of cognitive apprenticeship. Among other things, Vygotsky emphasized the sociocultural origins of higher mental functioning in individuals (Wertsch and Kanner, 1992) . He proposed the construct of the *zone of proximal development*, defined as the distance between a child's "actual development level as determined by independent problem solving" and the higher level of "potential development as determined through problem solving under adult guidance or in collaboration with more capable peers" (Vygotsky, 1978: 86) . Given the value of assisted performance in helping a student to make progress while in the zone, the utility of scaffolding is made manifest. A scaffold has five salient characteristics: "it provides a support; it functions as a tool, it extends the range of the worker; it allows the worker to accomplish a task not otherwise possible, and it is used selectively to aid the worker where needed" (Greenfield, 1984: 118) . These characteristics of a scaffold assist learners in navigating through the zone of proximal development.

While the method of cognitive apprenticeship is most readily adopted in the context of the classroom, the objective of our research is to attempt to realize these benefits in the context of computer-based learning environments as well. To this end, our approach includes: (a) situating learning so that conceptual and factual knowledge are exemplified in their contexts of use, (b) teaching processes that experts use so that students learn to "see" things the way experts see them, and (c) learning through guided experience that focuses on the development of metacognitive skills such as self-monitoring and self-correction. In terms of system design, cognitive apprenticeship requires the creation of appropriate activity settings characterized by good design and management of assisted performance (Gallimore and Tharp, 1990) . At the same time, we endeavor to provide system interfaces that support reflection (Collins and Brown, 1988) and the reification of abstract concepts and processes (Brown, 1985a) .

Overview of *SMALLTALKER*

SMALLTALKER is a multimedia, Macintosh-based interactive learning environment for Smalltalk programming. It is an 8-bit color system. The learning environment supports both the teaching of domain knowledge and the coaching of domain skill. We have made a concerted attempt to apply the methods of cognitive apprenticeship in designing and implementing the system. (For attempts by other researchers to apply cognitive apprenticeship to computer-based learning environments, see Lajoie & Lesgold (1989) , Miller (1992) , and Newman (1991) .) The start-up screen of *SMALLTALKER* is shown in Figure 1. In an attempt to convey an element of Smalltalk's history and culture, the start-up consists of animated balloons flying over a landscape that includes the Smalltalk castle (see *Byte Magazine*, August, 1981). The start-up is accompanied by music and played in a loop until a user moves the mouse. Moving the mouse initiates a dialog where the user is welcomed to the system and requested for information to provide a user profile.

After a digitized QuickTime™ movie of the instructor acquainting the student as to the purpose of and the functions available in *SMALLTALKER*, the system leads students through an orientation of its interface so that students can begin to interact effectively with the system. The use of various elements of the interface (eg. mouse, windows, and text manipulation) is demonstrated to students via movie clips. Students are then coached on the use of the interface in a different problem setting.

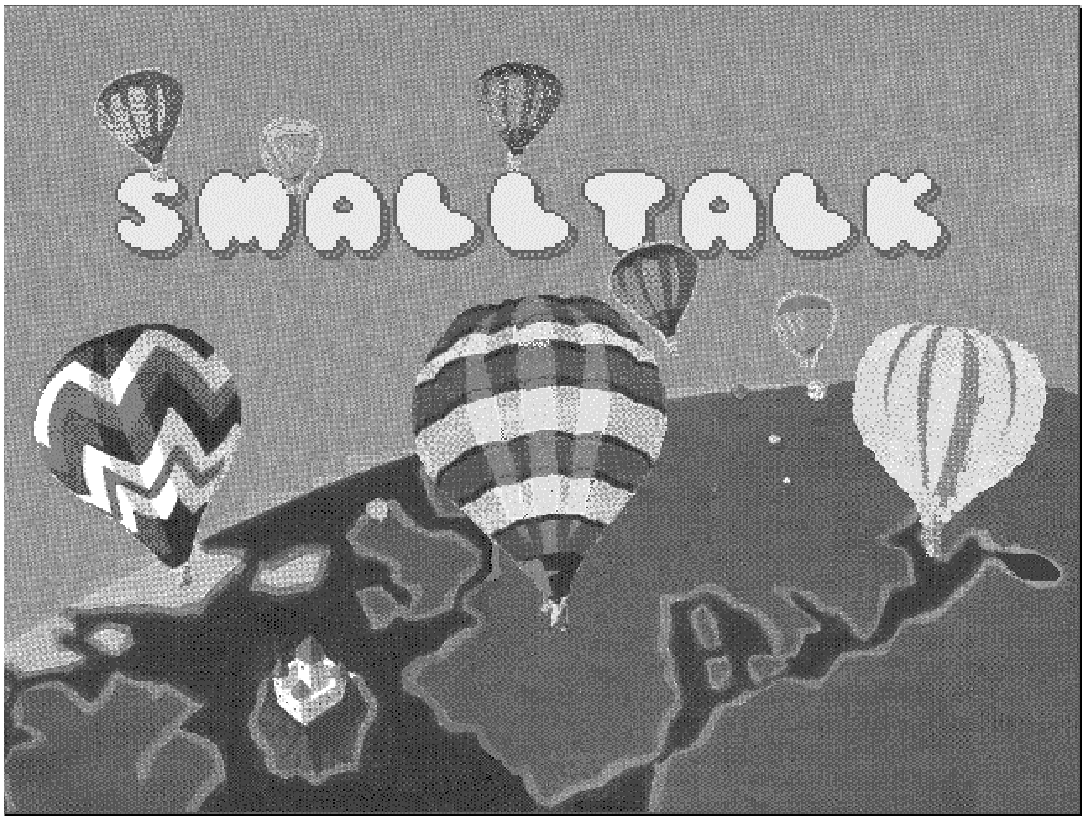
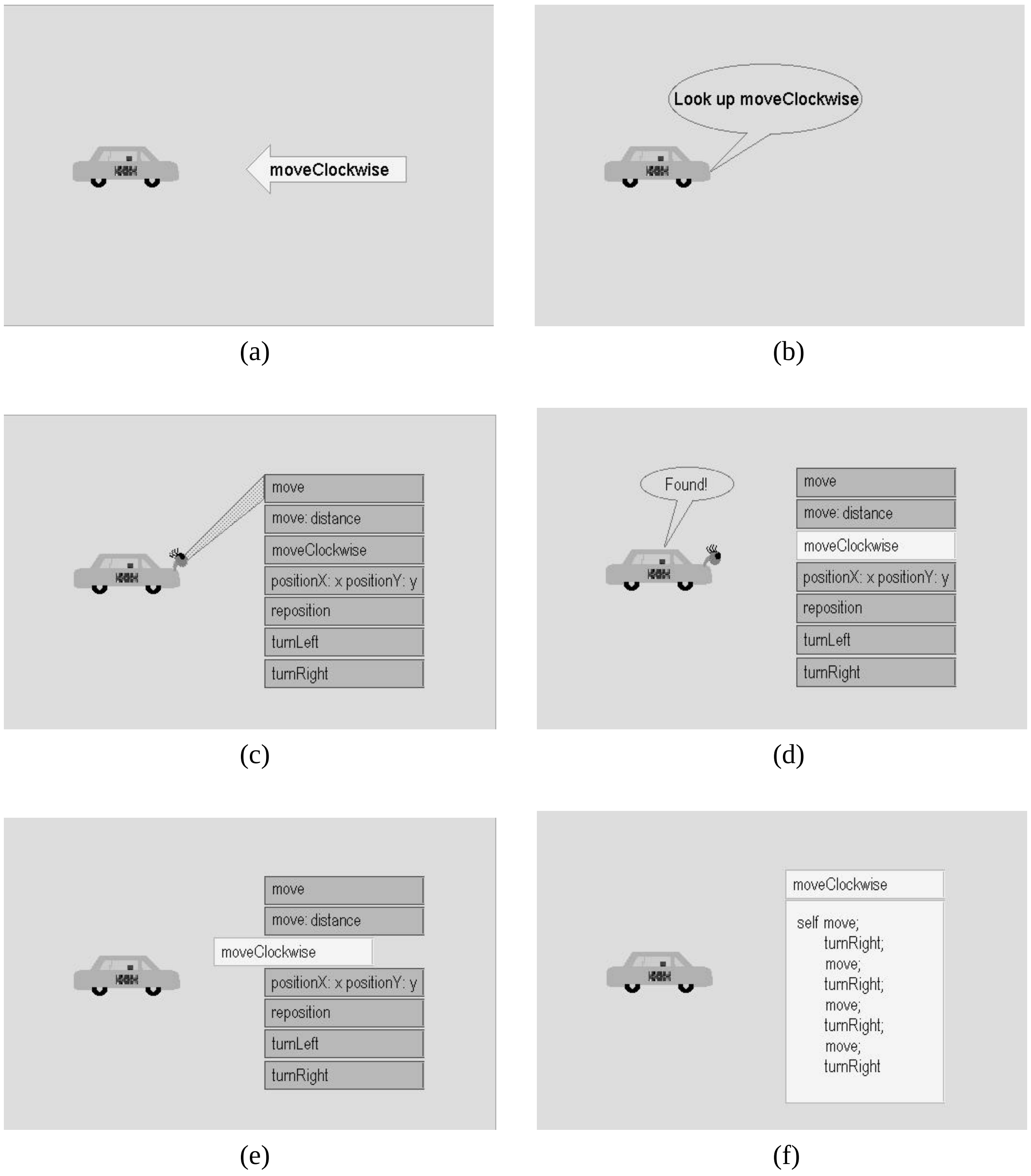


Figure 1. *SMALLTALKER* start-up screen

Having obtained basic interface skills, students are introduced to the important Smalltalk programming concept of message-passing. At this stage, students are taught to view object-oriented programming entirely in terms of objects communicating with one another by message passing. This view is referred to as an *external view* (LaLonde and Pugh, 1990b) where programming is treated entirely in terms of message-passing syntax (Shafer and Ritz, 1991) . Details of method implementation are hidden. The system's interaction with students is structured so as to reify the relationship between process and product. Thus, representations of objects that students interact with are shown graphically, and message passing invokes changes to objects that are visually representable.

After message passing has been learned, instance methods are introduced. This section opens up the *internal view* of Smalltalk programming (LaLonde and Pugh, 1990b) and its concomitant method-definition syntax (Shafer and Ritz, 1991) . Here, an important conceptual bridge must be established between the message received and the method executed. Figure 2 illustrates a sequence of frames that builds this bridge. The sequence focuses on the actions taken by an instance object, **TestCar**, when it receives the message **moveClockwise**. A search is carried out to determine whether there exists, from among the set of methods known to **TestCar**, a method corresponding to the name of the message received. As can be seen, the method **moveClockwise** is eventually found. The method is retrieved, and the instructions are displayed for execution. In this section, students also learn how to code instance methods, including methods with arguments, using a Simplified Browser.



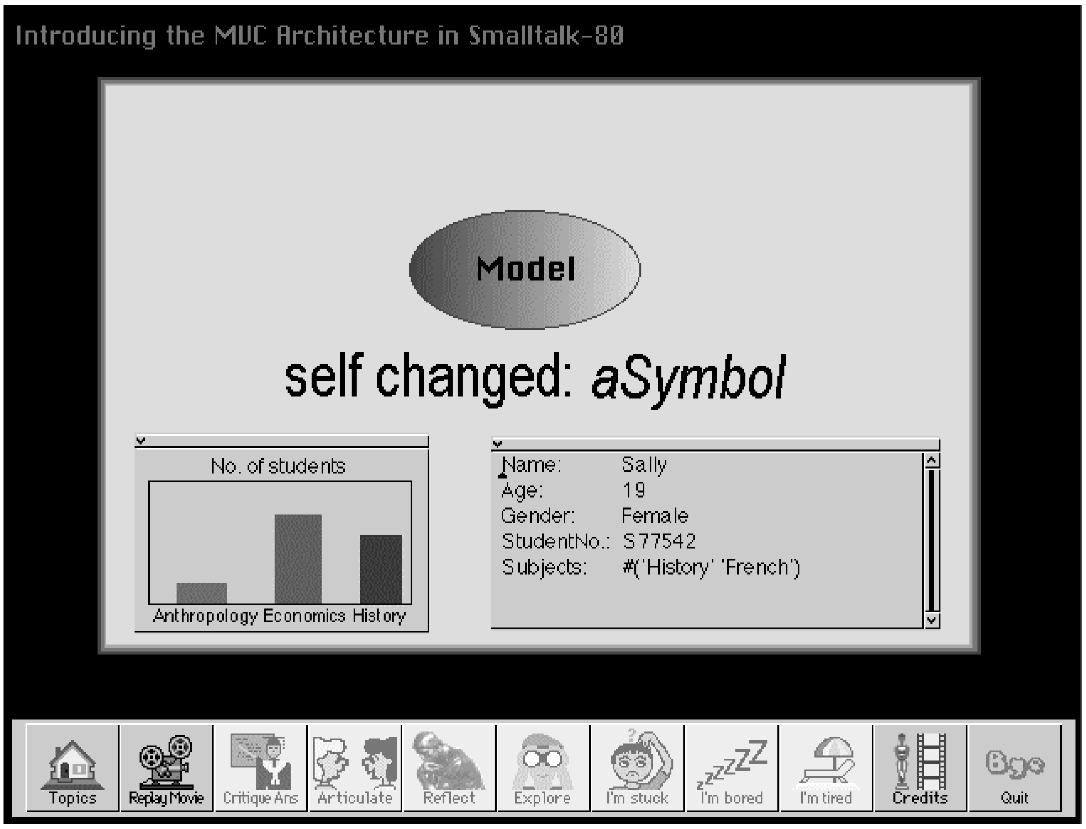
```
self move;
     turnRight;
     move;
     turnRight;
     move;
     turnRight;
     move;
     turnRight
```

Figure 2. Sequence of frames illustrating how **TestCar** searches for the instructions it must execute on receiving the message **moveClockwise**

The next topic covered is instance variables, the means by which values for features of instantiated objects are stored. The concept of assignment is also introduced at this time so that values can be assigned to instance variables.

Having learned about instance methods and instance variables, the distinction between instance objects and class objects is drawn. The topic of classes and class methods is presented, wherein students learn how a class object is used to create instance objects. They then move from using the Simplified Browser to a reduced version of Smalltalk–80's System Browser that allows the definition of instance as well as class methods. They also learn how to define a new class object. The instruction then progresses to the topic of class variables and other variables such as global variables and temporary variables.

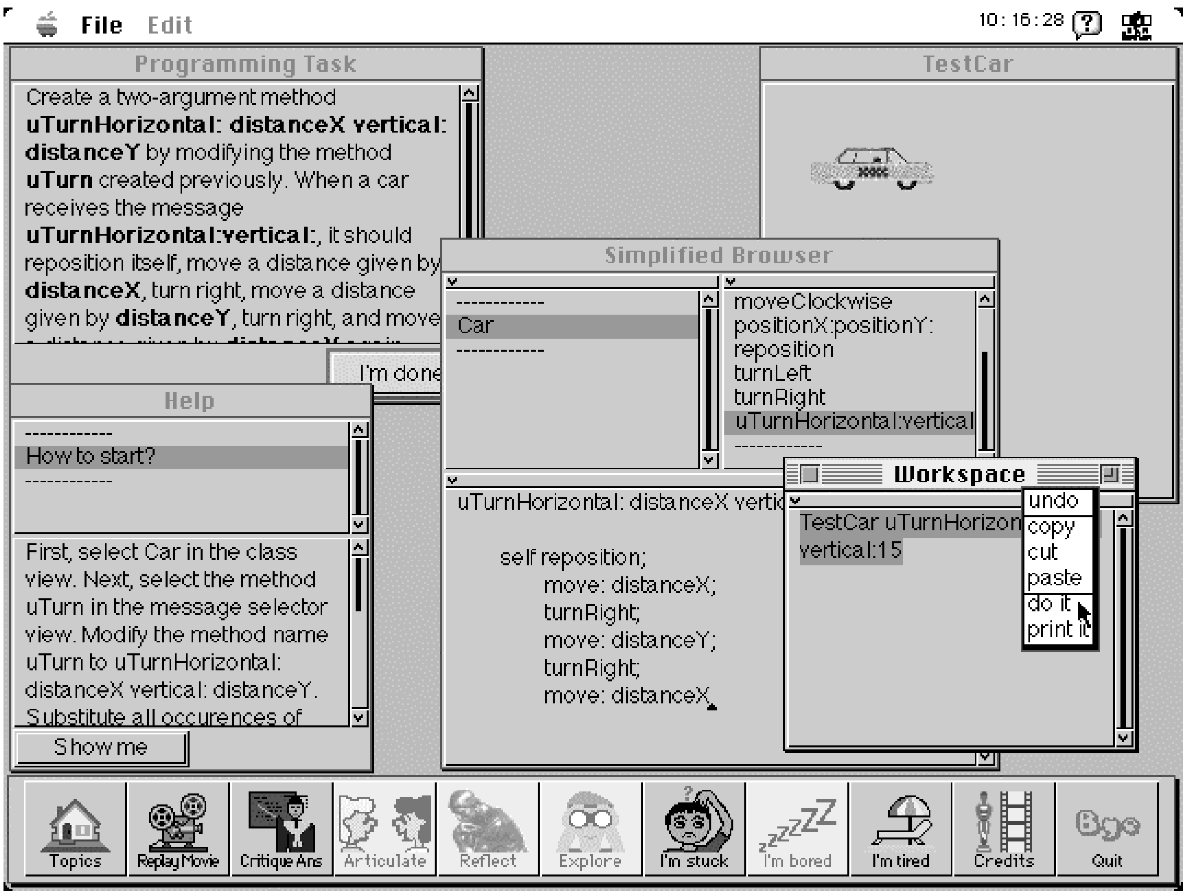
Next, students learn about superclasses and subclasses. In the process, they learn about the concept of inheritance and how it is used. They are also taught about the pseudo-variable **super**. Finally, students learn about the Model–View–Controller (MVC) architecture of Smalltalk–80 in the context of developing a simple windows-based database for inserting and deleting student records. In this embedded context, students learn how to create windows with associated controllers, and they learn about SelectionInList views, Text views, Dialog views, and buttons. Figure 3 illustrates how a Model notifies its dependent views that it has changed so that they can update their display. This concept is illustrated in the full context of the system's instruction presentation environment.



```
Name:        Sally
Age:         19
Gender:      Female
StudentNo.:  S77542
Subjects:    #('History' 'French')
```

Figure 3. Illustrating view dependency on Model in Smalltalk's MVC architecture

Apart from the instructional content, students are also required to complete programming tasks as they work through the different topics. Figure 4 shows a screen snapshot of a student working on a programming task. His task is to program a method, **uTurnHorizontal: distanceX vertical: distanceY**, to allow **TestCar** to reposition itself, then move forward, turn right, move forward, turn left, and move forward again. The snapshot illustrates that the student has coded the method in the Simplified Browser. In the Workspace, he has coded a message expression to test his method. He is about to click on "do it" in the pop-up menu to execute the selected message expression. When "do it" is selected, **TestCar** will execute the contents of the coded method in the graphic display window. This style of interaction is designed to provide immediate feedback through the user interface and to promote self-reflection should a coding error result.



Create a two-argument method **uTurnHorizontal: distanceX vertical: distanceY** by modifying the method **uTurn** created previously. When a car receives the message **uTurnHorizontal:vertical:**, it should reposition itself, move a distance given by **distanceX**, turn right, move a distance given by **distanceY**, turn right, and move

----
How to start?

First, select Car in the class view. Next, select the method uTurn in the message selector view. Modify the method name uTurn to uTurnHorizontal: distanceX vertical: distanceY. Substitute all occurences of

----
Car
----

moveClockwise
positionX:positionY:
reposition
turnLeft
turnRight
uTurnHorizontal:vertical

```
uTurnHorizontal: distanceX verti

    self reposition;
        move: distanceX;
        turnRight;
        move: distanceY;
        turnRight;
        move: distanceX.
```

```
TestCar uTurnHorizon
vertical:15
```

Figure 4. Snapshot of a programming task being worked on by a student

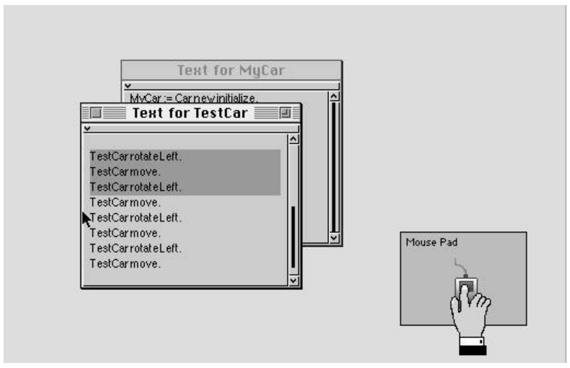
Applying cognitive apprenticeship in *SMALLTALKER*

In this section, we discuss the ways in which we have attempted to incorporate the instructional methods of cognitive apprenticeship in the design and implementation of *SMALLTALKER.*

Modeling

SMALLTALKER is unique in that it fulfills *two* roles: that of instruction presenter and that of coach. In its capacity as instruction presenter, *SMALLTALKER* makes extensive use of modeling. To support enculturation, the nurturing of domain-related values, QuickTime™ movies are used to show an expert presenting information to students. MacroMind Director™ animations are then used to present instruction on concepts and skills necessary for Smalltalk programming. The movies are accompanied by the expert's voice narration of the instruction.

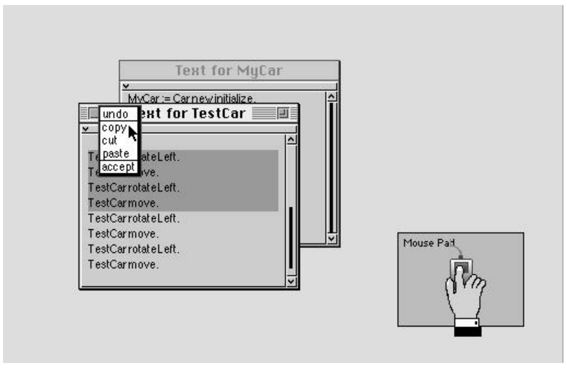
Figure 5 provides an illustration of how modeling occurs. The figure comprises four snapshots from a Director animation. In this example, the student is learning about the copy-and-paste technique in text editing. The snapshots show how the expert selects a portion of text (Figure 5a), selects "copy" from the pop-up menu (Figure 5b) resulting in the copied text appearing on a notional clipboard, then activating the window behind and, after clicking the mouse to establish the insertion point, selecting "paste" from the pop-up menu (Figure 5c), thus yielding the pasted text at the insertion point of the selected window (Figure 5d).



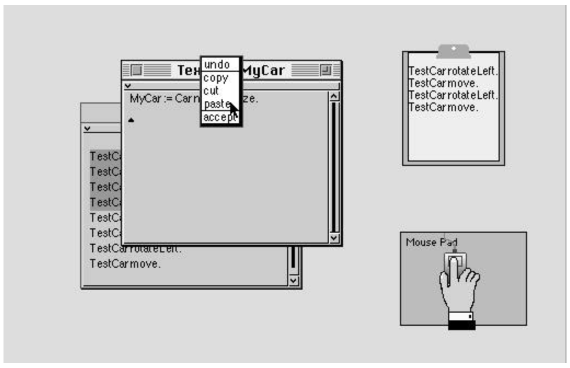
```
MyCar := Carnew initialize.

TestCarrotateLeft.
TestCarmove.
TestCarrotateLeft.
TestCarmove.
TestCarrotateLeft.
TestCarmove.
TestCarrotateLeft.
TestCarmove.
```

(a)



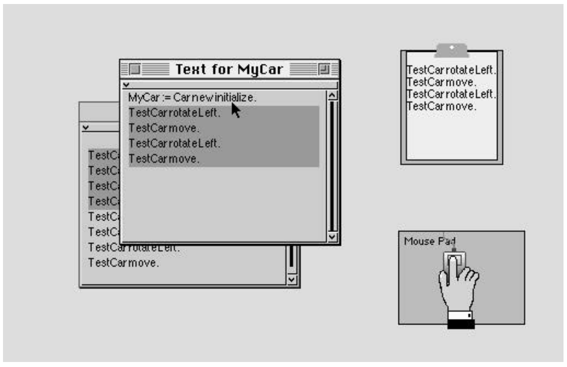
(b)



```
MyCar := Carnew initialize.
```

```
TestCarrotateLeft.
TestCarmove.
TestCarrotateLeft.
TestCarmove.
```

(c)



```
MyCar := Carnew initialize.
TestCarrotateLeft.
TestCarmove.
TestCarrotateLeft.
TestCarmove.
```

(d)

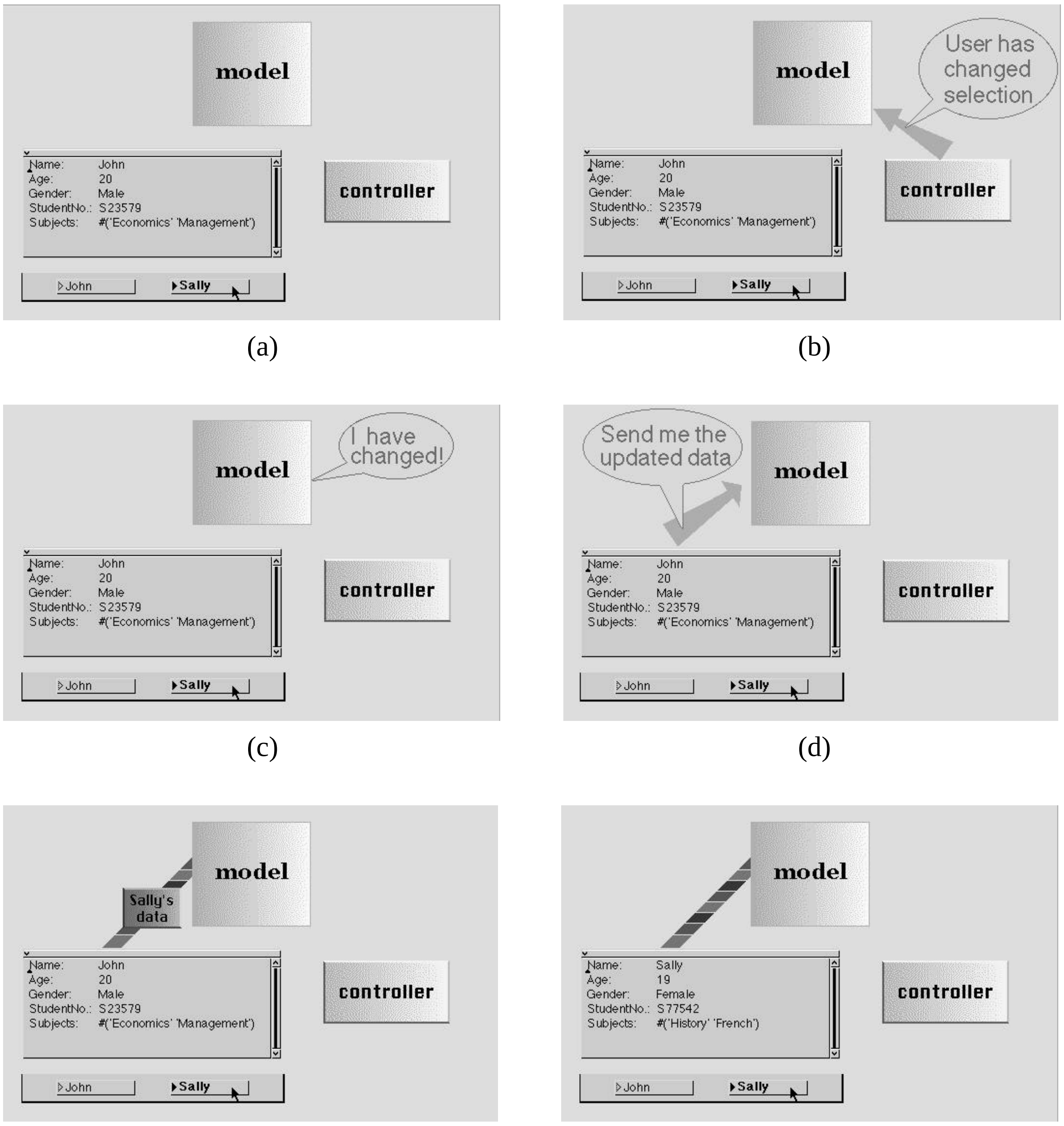
Figure 5. Modeling of copy-and-paste in text editing

Modeling is ideally suited to reifying processes through animation techniques. The above illustration depicts instruction on a physical process. Of greater importance, however, are conceptual processes that have no observable physical counterpart. The MVC architecture of Smalltalk–80 is, perhaps, the most difficult concept to grasp. Figure 6 shows how we use multimedia modeling to assist students in building a mental model of the Smalltalk processes that occur in the computer when the user selects the button "Sally". This approach promotes students' understanding of cause-and-effect relationships, an essential component for debugging knowledge in times of programming crisis. The use of multimedia allows concepts and processes to be "brought to life," focusing students' attention on the *meaning* and *sense* of what they observe. Our system tests with students confirm the effectiveness of multimedia-based modeling.

To further promote the understanding of cause-and-effect relationships, goals related to a sequence of actions are made explicit *before* the actions are shown, and the results of the actions are always shown in a concrete fashion. Our instruction is also designed to start with concrete and visual objects (eg. **TestCar**) before moving to more abstract and functional objects (eg. **MyRecord**) to aid students in grasping the necessary concepts.

We attempt to adhere to the advice of Collins *et al* (1989) with respect to the way sequencing of instruction and materials should be implemented. For example, increasing complexity is reflected in the way instance objects are introduced before class objects, and message passing is introduced before method definition. Similarly, Collins *et al* (1989) advise that global skills should be taught before local skills; that is, students should be allowed to build a conceptual model of how the pieces of a domain fit together before they attempt to produce the pieces themselves. Consequently, we teach students to *use* methods (in the form of messages) before teaching them to *code* methods.

Another prominent concern in cognitive apprenticeship is the use of authentic tasks. We find that authenticity has to be traded off somewhat in favour of an emphasis on achieving understanding. As authentic tasks are necessarily somewhat sizeable, it would require the student, with limited initial knowledge, to work largely "in the dark" to contribute to the whole task. We deem it inappropriate to compromise understanding for the sake of authentic tasks but have focused instead on supporting authentic *practice*. As an example, we teach the text-editing skill of copy-and-paste in the context of a realistic effort to modify code so that the skill learned will be relevant and useful in genuine programming situations.



(a) (b) (c) (d) (e)

Name: John
Age: 20
Gender: Male
StudentNo.: S23579
Subjects: #('Economics' 'Management')

(f)

Name: Sally
Age: 19
Gender: Female
StudentNo.: S77542
Subjects: #('History' 'French')

(a) (b)

(c) (d)

(e) (f)

Figure 6. Illustrating cause-and-effect relationships when the user clicks on the button "Sally"

Coaching

Coaching is a critical activity in any learning environment because it is here that the teacher or system engages in actions that will facilitate students' mastery of problem solving. In *SMALLTALKER*, students are made to work on programming problems while they are still being introduced to new concepts and procedures. The most pervasive method of coaching takes the form of feedback to student actions and errors while they are engaged on programming tasks. Unlike model tracing techniques often used in developing intelligent tutoring systems, *SMALLTALKER* does not intervene until students are ready to submit their programming effort to the system's scrutiny. As an example, a dialog box like that shown in Figure 7 will appear if a message expression coded by a student does not begin with the name of an object. Notice that the feedback given is highly situated; it is specific to the particular programming problem being solved.

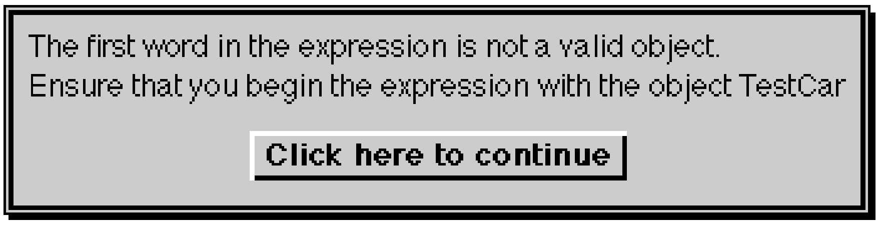

The first word in the expression is not a valid object.
Ensure that you begin the expression with the object TestCar

Figure 7. A typical feedback dialog

Besides the use of dialogs, feedback is also given by providing visual animation of objects where appropriate. As illustrated in Figure 4, the effect of executing the code written by a student is made self-evident from the animation of the car moving about in the **TestCar** window. Thus, students are encouraged to form and test hypotheses, diagnose their own performance, and adopt an active attitude to learning. In this way, students are trained to develop metacognitive skills. This approach also allows us, as system designers, to move away from attempting to make the system "intelligent" so that it can engage in diagnosis and remediation of student errors. Given the highly semantic nature of the Smalltalk programming language and the constrained nature of the programming tasks that students are asked to complete, the development of an "intelligent" automatic Smalltalk programming component and the use of student modeling turn out to be both difficult and unnecessary. Consequently, they were not attempted.

An important feature of cognitive apprenticeship is that the efforts of students or novices should be useful as well as used. To this end, problem solving tasks often require students to make enhancements to existing methods or to add new methods to existing code so that the resulting whole possesses greater value after such changes.

We attempt to control interface complexity by getting students to begin coding message expressions in a customized Workspace with basic functionality. Similarly, students are introduced to instance objects and instance methods in the context of a Simplified Browser. In this way, the full complexity of the standard Smalltalk System Browser is hidden until more sophisticated functions are needed.

We also follow the principle of increasing complexity when engaging in coaching. As an illustration, students attempt programming tasks in the following sequence: sending unary messages, executing a message expression series, sending keyword messages (with students allowed discretion in choosing argument values), sending more complex messages involving precedence, creating a method, and creating a class. In addition, as more concepts and skills are mastered, the sophistication of programming problems increases, thereby demanding an increasing diversity of skills from the student together with the ability to integrate those skills.

Scaffolding and fading

Scaffolding and fading are difficult instructional methods to implement because they require a teacher or system to be sensitive to the specific needs and difficulties of students engaged in task performance at any particular point in time. Our solution has been to place the burden of responsibility on the shoulders of students themselves. *SMALLTALKER* provides a fairly open-ended help system (see Figure 4) that students can activate by selecting the **I'm stuck** button in the learning environment. The help window lists a set of topics on which assistance on specific aspects of the programming task students are working on is available. Making the activation of the help system the responsibility of students encourages them to take active control of their learning experience. This approach is in keeping with the spirit of cognitive apprenticeship and its concern with making students grow out of dependency on the instructor so that they can be independent thinkers.

The learning environment also contains a **Replay Movie** button. Students can select this button if they wish to review the instructional content related to the problem. Should they require material from an earlier instructional module, they can access these topics through the **Topics** button in the learning environment. A set of access points to the different modules is presented in the form of jigsaw peices that students can select from. Selecting a jigsaw piece leads to a further animated segmentation of that piece, showing subtopics that students can select. In this manner, all instructional material is readily accessible.

The **Show me** button in the help window is particularly significant. It functions as a student's last course of action. For this reason, it is separated from the help topic list. The **Show me** button is not always available. We have implemented the button such that it only becomes available after the student has selected some percentage of the help items available. This approach prevents students from choosing the easy way out of a problem solving situation. Designing the help system in this way enforces a more active learning approach on the part of students. When students eventually select the **Show me** button, a movie will play, showing how an expert would have completed the programming task. This action effectively reverts back to the modeling method of instruction.

Programming exercises may entail placing students in the zone of proximal development. For example, one programming exercise in the section on message passing requires the coding of the expression

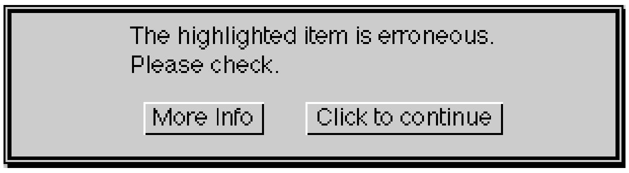
MyRectangle area + MyTriangle area

However, the binary message selector **+** (addition) had not been introduced previously. Only the binary message selector ***** (multiplication) had been encountered before, but in a different context: that of the keyword message

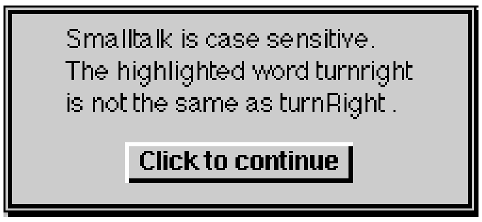
TestCar move: 20*2

As the question requires knowledge that had never been presented before, it contains a hint to the effect that unary messages are evaluated *before* binary messages. Cognitively, the programming task is more demanding because it involves greater complexity in the precedence of evaluation. Consequently, scaffolding is provided through the help system to aid students in traversing the zone of proximal development with respect to that problem.

Fading has been implemented in the kind of error trapping that the system engages in. During the course of early problem solving, special checks are incorporated to prevent students from committing errors from which they would be unable to recover. However, as students become more familiar with Smalltalk programming, feedback dialogs are generalized so as to provide less problem solving support. The rationale, once again, is that students should begin to think for themselves and take responsibility for their own learning. Figure 8a illustrates a feedback dialog with fading, while Figure 8b illustrates a dialog without fading. However, even with fading in operation, students have the option of requesting a more specific dialog if they feel that the level of support received is inadequate in their particular circumstance. All they have to do is click on the button **More Info**, and a more detailed level of dialog (as in Figure 8b) will be shown.


The highlighted item is erroneous.
Please check.

(a)


Smalltalk is case sensitive.
The highlighted word turnright
is not the same as turnRight .

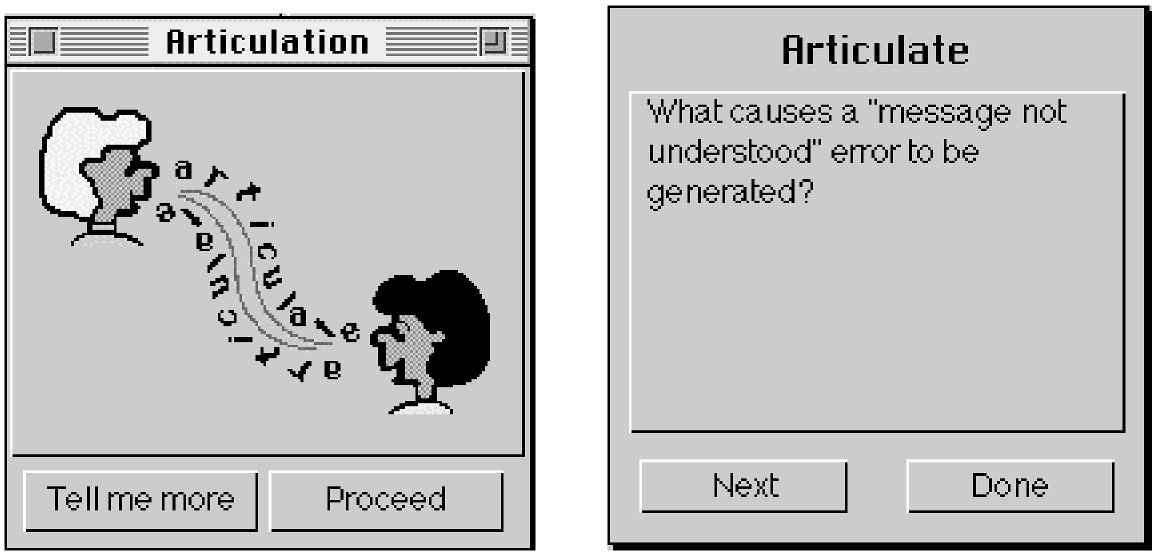
(b)

Figure 8. Fading in feedback dialogs. (a) More general (b) More specific

Articulation

Computers are ill-suited to handling the natural language articulations of humans. Nevertheless, we encourage students to articulate their knowledge by posing questions that request them to articulate answers to various conceptual questions, either to themselves or to a friend. The **Articulate** button becomes active within the learning environment as soon as a student completes a portion of instructional material that an articulation question is based on. The student does not have to click on the **Articulate** button immediately. As he progresses through the instructional material, more articulation questions are added to *SMALLTALKER*'s list of questions that can be presented to the student. Consequently, the number of articulation questions available to a student at any point in time will depend on how far he has progressed through the instructional module and whether he has chosen the **Articulate** button and answered some articulation questions before.

When a student clicks the **Articulate** button in the learning environment, an Articulation window appears (see Figure 9a). Clicking on the button **Tell me more** plays a QuickTime™ movie of the instructor explaining the rationale of articulation as an instructional method. Clicking on the button **Proceed** takes the student to a specific question the answer to which requires articulation (see Figure 9b).



What causes a "message not understood" error to be generated?

(a) (b)

Figure 9. Illustration of the teaching method articulation

For illustrative purposes, Table 1 lists the entire set of articulation questions for the instructional module on superclasses and subclasses, divided into the module's subtopics.

Table 1. Articulation questions on superclasses and subclasses

Introducing superclass

1. What is the relationship between a superclass and a subclass?
2. How do you determine the superclass of a new class that you are going to define?

Making use of inheritance

1. Describe the property of inheritance. What different types of items participate in inheritance?

Creating and using a more complex subclass

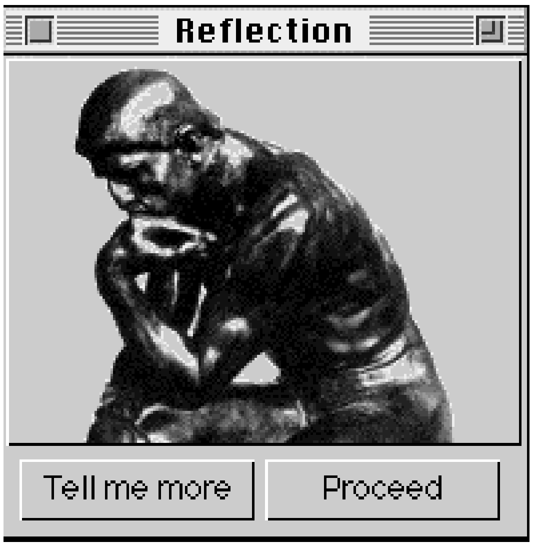
1. What is a bag? Why would you use a bag rather than an array?

Understanding the pseudo-variable super

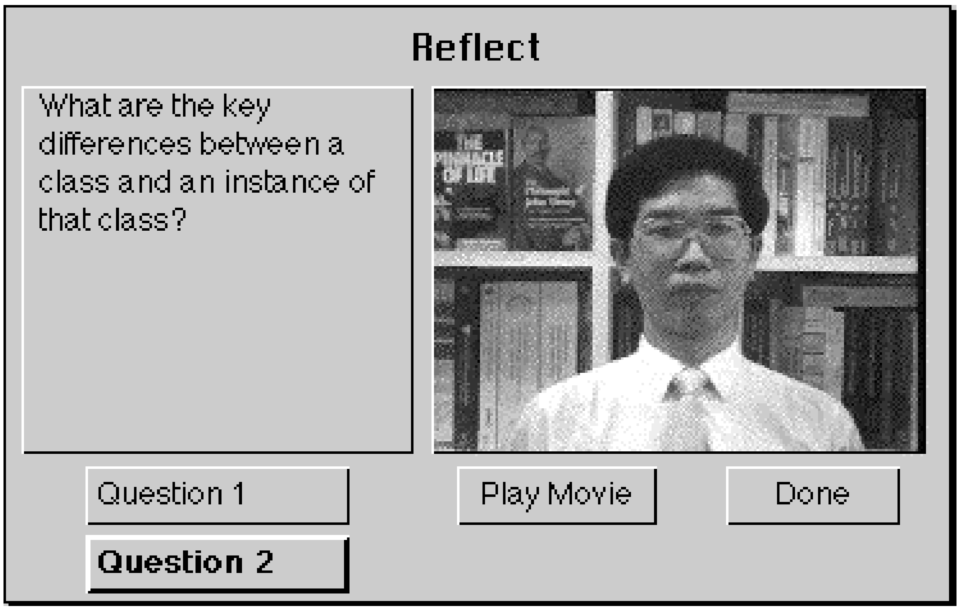
1. What are the differences between the pseudo-variables **super** and **self**?
2. What situation can cause an infinite loop when the method **new** is invoked?
3. In a situation where a method of the same name is defined for every level up an inheritance chain, explain how Smalltalk determines which particular method should be executed when the pseudo-variable **super** is used.

Reflection

We attempt to provoke reflection in students by periodically posing questions that either (a) possess deeper conceptual significance, or (b) involve subtleties related to programming practice. Reflection questions can be called up by clicking on the button marked **Reflect** in the learning environment. When a student clicks on the button **Reflect**, a Reflection window appears, containing two buttons (see Figure 10a). Clicking on the **Tell me more** button plays a QuickTime™ movie of the instructor explaining the purpose of reflection. Clicking on the **Proceed** button takes the student to a specific question the answer to which requires reflection (see Figure 10b). After students have spent some time reflecting on the question posed, they can click on the button **Play Movie** to listen to an expert expressing his view on the question. In this way, students can gauge to what extent they have come to appreciate the subject domain in the way that an expert does.



(a)



What are the key differences between a class and an instance of that class?

(b)

Figure 10. Reflection in *SMALLTALKER*

Exploration

It turns out that exploration is an activity that system designers cannot prevent students from engaging in if the system succeeds in gripping their interest, exciting positive emotions, and arousing their motivation to learn. We support students' exploration of Smalltalk programming by providing them with an **Explore** button in the learning environment. Clicking on this button takes students into the ParcPlace Smalltalk–80 programming environment where they can proceed to explore and experiment with system classes not covered in *SMALLTALKER*'s instructional modules. Students can then proceed to pursue their own goals and develop their own applications when they have mustered enough confidence to do so. A **Return to *SMALLTALKER*** button allows students to return to the *SMALLTALKER* learning environment.

Table 2 summarizes the teaching methods of cognitive apprenticeship and the ways in which these teaching methods have been instantiated in *SMALLTALKER*.

Table 2. Cognitive apprenticeship teaching methods and their instantiation in *SMALLTALKER*

Cognitive apprenticeship teaching methods	Instantiation in *SMALLTALKER*
Modeling	Expert communicates with student via digitized video
	Expert shows how things work and how things are done using animations accompanied by voice narration
	Expert reifies cause-and-effect relationships; presents goals before actions
Coaching	Students work on programming tasks of increasing difficulty
	Highly situated feedback is given in response to student errors and actions
Scaffolding and fading	Student-initiated help system available through the **I'm stuck** button
	Students can replay movies to review instructional materials
	Help system provides a **Show Me** button as a last recourse
	Feedback dialogs are generalized when errors of the same type are made but recourse to more detailed information remains available
Articulation	Conceptual questions are posed to students asking them to articulate the answers to the questions either to themselves or to a friend
Reflection	Questions possessing deeper conceptual significance or that involve subtleties related to programming practice are posed to students to reflect upon
	Play Movie button plays a digitized movie of an expert expressing his view on the reflection question posed
Exploration	**Explore** button places students in the Smalltalk–80 programming environment so that they can further explore the system on their own and pursue their own goals

Discussion

In this section, we compare the *SMALLTALKER* research effort with work done by Carroll, Rosson, and their colleagues at the IBM T. J. Watson Research Center on minimalist instruction for Smalltalk programming. We then discuss the following issues related to our work on the *SMALLTALKER* project: (a) development time and effort, (b) results of informal formative evaluation, (c) plans for summative evaluation, (d) critique of the system's perceived limitations, and (e) plans for related work.

For several years, Carroll, Rosson, and their colleagues have investigated the learning of Smalltalk programming in the context of MiTTS— Minimalist Tutorial and Tools for Smalltalk (Carroll *et al*, 1990; Rosson *et al*, 1990). In addition to a minimalist tutorial comprising five instructional blocks designed around the game BlackJack, MiTTS has two software components. The first component is the Bittitalk Browser, a stripped-down Class Hierarchy Browser that provides a filtered view of classes and methods. The second component is the View Matcher, a structured browser that displays five coordinated views: the Application View, the Stack View, the Class Hierarchy View, the Inspector View, and the Commentary View. The View Matcher allows an application to be browsed as it is run. Hence, MVC dynamics can be interactively browsed and analyzed.

The goals of Carroll and Rosson's work are essentially twofold: first, to streamline and organize the content of manuals, tutorials, and other educational artifacts according to a minimalist instruction philosophy (Carroll *et al*, 1990); second, to support users in accomplishing real and meaningful tasks quickly while allowing them to take advantage of existing task knowledge in learning about a new system. The work on MiTTS thus shares certain similarities with the *SMALLTALKER* effort. For example, both emphasize real and meaningful programming tasks (authentic programming tasks using the terminology of cognitive apprenticeship). Both approaches also feature a reduced version of the Class Hierarchy Browser: MiTTS uses the Bittitalk Browser while *SMALLTALKER* uses the Simplified Browser.

Apart from the above similarities, however, MiTTS and *SMALLTALKER* are more different than alike. Most significantly, MiTTS is intended for use by experienced procedural programmers with some awareness of object-oriented concepts (Carroll *et al*, 1990). Consequently, it does not address learning of the syntax and semantics of Smalltalk *per se*. Instead, it focuses on the application level (the game BlackJack) and attempts to help the learner to understand the MVC paradigm of Smalltalk programming through the provision of View Matcher's five coordinated views. The learning process supported is open-ended, highly exploratory, and requires intrinsic self-motivation on the part of the learner.

By contrast, *SMALLTALKER* is targeted at programming novices with little prior programming knowledge. Consequently, the system takes students in a step-by-step fashion from the most elementary topic (Interface Orientation) to the most advanced topic (the MVC Architecture). Programming novices faced with View Matcher would be unable to proceed in any meaningful way; they would also be completely overwhelmed by the complexity of the View Matcher interface. Rosson *et al* (1990: 428) state: "From the start, we observed that learners had difficulty with Smalltalk's interaction techniques. The mechanics of scrolling or re-sizing windows often overwhelmed the exploration of BlackJack." It appears, then, that the decision to commence with an orientation to the system's interface in *SMALLTALKER* is sound. Finally, although Rosson *et al* (1990) eschew a linearly structured curriculum, they found that the complexity and novelty of the Smalltalk domain compelled their use of such a curriculum. In targeting an audience of novice programmers, we have found that there is no other course.

As with all large-scale design efforts, conceptualization of the *SMALLTALKER* system involved an extended iterative effort proceeding through many cycles of prototyping. Once the design was largely crystallized, however, development effort for both the instructional materials and system programming took approximately three-and-a-half man-years. The design of the Smalltalk instruction set (instructions as well as programming tasks) is, of course, predicated upon a good understanding of the Smalltalk programming language. The instruction set went through many iterations of refinement and informal testing to ensure that the materials presented and the way in which they were presented made them comprehensible to the target audience of entry-level university students.

Our evaluation of *SMALLTALKER* has, so far, been encouraging. In general, students have found the learning environment very user-friendly and generally supportive of their learning needs. In preliminary testing of *SMALLTALKER* with six students, we often found them engaging in activities that we did not anticipate. For example, during an early stage of learning when we intended students to select and execute single message expressions one at a time, we found some students taking the initiative to select a series of message expressions and trying to execute the series all at once. All would have been well were it not for the fact that message expressions in a series need to be separated by periods. We learned that the challenge to us, as system designers, is to provide the flexibility necessary to allow potentially rewarding explorations without confining students' activities within a straitjacket.

The development of *SMALLTALKER* is nearly complete. We are presently engaged in integration testing and ironing out bugs in the controls implemented for MVC programming tasks. Upon completion of this work, we intend to deploy the system within our Department for the purpose of field testing. Three summative evaluation studies have also been planned. The first study involves a simple two-factor study directed principally at establishing the effectiveness of learning Smalltalk using *SMALLTALKER* versus a more conventional teaching method. Affective dimensions of system use will also be evaluated. By virtue of its adherence to classical research techniques, this study is of necessity precise, but coarse. To provide a complement to the first evaluation, the second study will make use of a microgenetic analysis of a collaborating pair of students learning Smalltalk programming with the system. All learning sessions will be video-recorded, and the videotapes will be transcribed and analyzed. We fully recognize the confounding effect (within the classical research tradition) of introducing a second student into the learning setup. However, this study is designed to study processes of conceptual development and conceptual change in learning situations that incorporate instructional technology. Hence, the study conforms to a quite different tradition of analysis (see Chee *et al*, 1993; Roschelle, 1992) . Finally, the third study will involve one-to-one observation of a small number of students using *SMALLTALKER* individually. The aim of this study is to establish patterns of behavior in the use of the system and to systematically identify the strengths as well as weaknesses and limitations of *SMALLTALKER* in a realistic context of use. The three studies outlined constitute major research efforts in their own right. Research findings will be published in due course.

While the *SMALLTALKER* system appears generally effective in relation to the purpose for which it was designed, we feel uncertain about the instructional module on MVC. This topic is well-known among Smalltalk programmers for its complexity and difficulty. Mastery of the topic is acquired only with extended use and learning from mistakes while on the path of learning. Consequently, compressing the topic into an instructional module of a learning environment, we feel, imposes unreasonable demands on the system. This situation probably manifests a limitation that arises from the way instructional technology is being employed in this instance. Instructional technology cannot produce learning miracles. We believe that, in such situtations, a broader, social learning environment that involves collaboration between individuals and in which instructional technology is just one component of the whole learning milieu is needed for effective learning.

Finally, it should be recognized that *SMALLTALKER* only attempts to provide students with a first course in Smalltalk programming. Given the complexity of Smalltalk programming and the need to support the development of programming skills as part of a broader community of practice, we have embarked on the development of a complementary learning environment *CALET*, the *C*ognitive *A*pprenticeship *L*earning *E*nvironmen*T*, that has as its objective the provision of situated Smalltalk programming cases for realistic "on-the-job" programming practice.

Conclusion

In this paper, we motivated the need for more effective approaches to education and learning because an urgent need exists to develop instructional methods that can foster deep and robust learning relevant and applicable to real-world, non-school situations. To this end, we explored the use of cognitive apprenticeship in the domain of learning Smalltalk programming. We described our efforts in implementing cognitive apprenticeship in *SMALLTALKER* and illustrated how we instantiated the instructional methods of modeling, coaching, scaffolding and fading, articulation, reflection, and exploration. We also highlighted some of the difficulties and tradeoffs involved in system design. Not least, we discussed the issue of system evaluation and provided an informal critique of the system's perceived limitations. Despite these perceived limitations, our experience with *SMALLTALKER* makes us guardedly optimistic that such systems, when designed sensitively and used judiciously, can make a positive contribution to the goals of education.

Acknowledgements

The author is grateful to Jee Teck Tan, research assistant, for the effort devoted to implementing the *SMALLTALKER* system. Funding for this research project was provided by the National University of Singapore under research grant RP890618.

References

Allen, C. L. (1992). Multimedia learning environments designed with organizing principles from non-school settings. In E. De Corte, M. C. Linn, H. Mandl and L. Verschaffel (Eds.), *Computer-Based Learning Environments and Problem Solving* (pp. 465–484). Berlin: Springer-Verlag.

Bransford, J. D., Franks, J. J., Vye, N. J. and Sherwood, R. D. (1989). New approaches to instruction: because wisdom can't be told. In S. Vosniadou and A. Ortony (Eds.), *Similarity and Analogical Reasoning* (pp. 470-497). NY: Cambridge University Press.

Brown, A. L. and Palincsar, A. S. (1989). Guided, cooperative learning and individual knowledge acquisition. In L. Resnick (Ed.), *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser* (pp. 393-451). Hillsdale: NJ: Lawrence Erlbaum Associates.

Brown, J. S. (1985a). Idea amplifiers: New kinds of electronic learning environments. *Educational Horizons, 63*, 108-112.

Brown, J. S. (1985b). Process versus product: A perspective on tools for communal and informal electronic learning. *Journal of Educational Computing Research, 1*(2), 179–201.

Brown, J. S. (1990). Toward a new epistemology for learning. In C. Frasson and G. Gauthier (Eds.), *Intelligent Tutoring Systems: At the Crossroads of Artificial Intelligence and Education* (pp. 266-282). Norwood, NJ: Ablex.

Brown, J. S., Collins, A. and Duguid, P. (1988). Situated Cognition and the Culture of Learning. IRL Report No. 88-0008, Institute for Research on Learning.

Burger, M. L. and DeSoi, J. F. (1992). The cognitive apprenticeship analogue: a strategy for using ITS technology for the delivery of instruction and as a research tool for the study of teaching and learning. *International Journal of Man-Machine Studies, 36*, 775-795.

Carroll, J. M. (1990). *The Nurnberg Funnel: Designing Minimalist Instruction for Practical Computer Skill*. Cambridge, MA: MIT Press.

Carroll, J. M., Singer, J. A., Bellamy, R. K. E., & Alpert, S. R. (1990). A view matcher for learning Smalltalk. In Proceedings of the CHI ‘90 Conference on Human Factors in Computing Systems (pp. 431–437). NY: ACM.

Chapman, M. (1988). *Constructive Evolution: Origins and Development of Piaget’s Thought.* New York: Cambridge University Press.

Chee, Y. S., Tan, J. T. and Chan, T. (1993). Situated sense-making: A study of conceptual change in activity-based learning. In Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society, Boulder, CO. Hillsdale, NJ: Lawrence Erlbaum.

Clancey, W. J. (1991a). The frame of reference problem in the design of intelligent machines. In K. VanLehn (Ed.), *Architectures for Intelligence* (pp. 357-423). Hillsdale, NJ: Lawrence Erlbaum.

Clancey, W. J. (1991b). Israel Rosenfield, *The Invention of Memory: A New View of the Brain. Artificial Intelligence, 50*, 241-284.

Clancey, W. J. (1991c). Situated cognition: Stepping out of representational flatland. *AICOM, 4*(2/3), 109-112.

Clancey, W. J. (1992a). New perspectives on cognition and instructional technology. In E. Costa (Ed.), *New Directions for Intelligent Tutoring Systems* (pp. 3–14). Berlin: Springer-Verlag.

Clancey, W. J. (1992b). Representations of knowing: In defense of cognitive apprenticeship. *Journal of Artificial Intelligence in Education, 3*, 139-168.

Clancey, W. J. and Roschelle, J. (1991). Situated cognition: How representations are created and given meaning. In Proceedings of the AERA Symposium, Implications of Cognitive Theories of How the Nervous System Functions for Research and Practice in Education, Chicago. AERA.

Collins, A. (1991). Cognitive apprenticeship and instructional technology. In L. Idol and B. F. Jones (Eds.), *Educational Values and Cognitive Instruction: Implications for Reform* (pp. 121–138). Hillsdale, NJ: Lawrence Erlbaum.

Collins, A. and Brown, J. S. (1988). The computer as a tool for learning through reflection. In H. Mandl and A. Lesgold (Eds.), *Learning Issues for Intelligent Tutoring Systems* (pp. 1-18). Berlin: Springer-Verlag.

Collins, A., Brown, J. S. and Newman, S. E. (1989). Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In L. B. Resnick (Ed.), *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser* (pp. 453-494). Hillsdale, NJ: Lawrence Erlbaum.

Edmondson, W. (Ed.) (1985). *The Age of Access: Information Technology and Social Revolution, Posthumous Papers of Clin Cherry*. London: Croon Helm.

Feinstein, D. (1988, Winter). Learning how to learn. *Benchmark*, pp. 2–5.

Forman, G. and Pufall, P. B. (Eds.) (1988). *Constructivism in the Computer Age*. Hillsdale, NJ: Lawrence Erlbaum.

Gallimore, R. and Tharp, R. (1990). Teaching mind in society: Teaching, schooling, and literate discourse. In L. C. Moll (Ed.), *Vygotsky and Education: Instructional Implications and Applications of Sociohistorical Psychology* (pp. 175-205). NY: Cambridge University Press.

Greenfield, P. M. (1984). A theory of the teacher in the learning activities of everyday life. In B. Rogoff and J. Lave (Eds.), *Everyday Cognition: Its Development in Social Context* (pp. 117-138). Cambridge, MA: Harvard University Press.

Harel, I. and Papert, S. (Eds.) (1991). *Constructionism.* Norwood, NJ: Ablex.

Jonassen, D. H. (1991, September). Evaluating constructivistic learning. *Educational Technology,* pp. 28–33.

Lajoie, S. P. and Lesgold, A. (1989). Apprenticeship training in the workplace: Computer-coached practice environment as a new form of apprenticeship. *Machine-Mediated Learning, 3,* 7-28.

LaLonde, W. and Pugh, J. (1990a, Novermber/December). Smalltalk as a first programming language: the Carleton experience. *Journal of Object-Oriented Programming,* pp. 60-65.

LaLonde, W. R. and Pugh, J. R. (1990b). *Inside Smalltalk, Volume 1.* Englewood Cliffs, NJ: Prentice-Hall.

Lave, J. (1988). The Culture of Acquisition and the Practice of Understanding. IRL Report No. 88-0007, Institute for Research on Learning.

Lave, J. and Wenger, E. (1991). *Situated Learning: Legitimate Peripheral Participation.* NY: Cambridge University Press.

Lawrence, J. A. and Valsiner, J. (1993). Conceptual roots of internalization: From transmission to transformation. *Human Development, 36,* 150–167.

Miller, A. L. (1992). An intelligent tutoring system prototype based on cognitive apprenticeship. In Proceedings of the ITS'92 Conference, Montréal, Canada.

Newman, D. (1991). Interpreting an intelligent tutor's algorithmic task: A role for apprenticeship as a model for instructional design. *AI and Society, 5,* 93-109.

Nielsen, J. and Richards, J. T. (1989). The experience of learning and using Smalltalk. *IEEE Software, 6*(3), 73-77.

Paris, S. G. and Winograd, P. (1990). How metacognition can promote academic learning and instruction. In B. F. Jones and L. Idol (Eds.), *Dimensions of Thinking and Cognitive Instruction* (pp. 15–51). Hillsdale, NJ: Lawrence Erlbaum.

Pea, R. D. (1991). Learning through multimedia. *IEEE Computer Graphics and Applications, 11*(4), 58–66.

Pea, R. D. (1992a). Augmenting the discourse of learning with computer-based learning environments. In E. De Corte, M. C. Linn, H. Mandl and L. Verschaffel (Eds.), *Computer-Based Learning Environments and Problem Solving* (pp. 313–343). Berlin: Springer-Verlag.

Pea, R. D. (1992b). Distributed multimedia learning environments: Why and how? *Interactive Learning Environments, 2*(2), 73–109.

Resnick, L. B. (1987). Learning in school and out. *Educational Researcher, 16*(9), 13-20.

Rogoff, B. and Lave, J. (Eds.) (1984). *Everyday Cognition: Its Development in Social Context.* Cambridge, MA: Harvard University Press.

Roschelle, J. (1992). Learning by collaborating: Convergent conceptual change. *The Journal of the Learning Sciences, 2*(3), 235–276.

Roschelle, J. and Clancey, W. J. (1992). Learning as social and neural. *Educational Psychologist, 27*(4), 435–453.

Rosson, M. B., Carroll, J. M., Bellamy, R. K. E. (1990). Smalltalk scaffolding: A case study of minimalist instruction. In Proceedings of the CHI ‘90 Conference on Human Factors in Computing Systems (pp. 423–429). NY: ACM.

Sandberg, J. and Wielinga, B. (1992). Situated cognition: A paradigm shift? *Journal of Artificial Intelligence in Education, 3,* 129-138.

Scardamalia, M. and Bereiter, C. (1991). Higher levels of agency for children in knowledge building: A challenge for the design of new knowledge media. *The Journal of the Learning Sciences, 1*(1), 37-68.

Scardamalia, M., Bereiter, C. and Steinbach, R. (1984). Teachability of reflective processes in written composition. *Cognitive Science, 8*, 173-190.

Schank, R. C. and Edelson, D. J. (1989/90). A role for AI in education: Using technology to reshape education. *Journal of Artificial Intelligence in Education, 1*(2), 3-20.

Schank, R. C. and Jona, M. Y. (1991). Empowering the student: New perspectives on the design of teaching systems. *The Journal of the Learning Sciences, 1*(1), 7-35.

Shafer, D. and Ritz, D. A. (1991). *Practical Smalltalk: Using Smalltalk/V.* NY: Springer-Verlag.

Smith, J. P., diSessa, A. A. and Roschelle, J. (1993). Misconceptions reconceived: A constructivist analysis of knowledge in transition. *The Journal of the Learning Sciences, 3*(2), 115–163.

Still, A. and Costall, A. (Eds.) (1991). *Against Cognitivism*. London: Harverster Wheatsheaf.

Vygotsky, L. S. (1978). *Mind in Society: The Development of Higher Psychological Processes*. Cambridge, MA: Harvard University Press.

Vygotsky, L. S. (1987). *The Collected Works of L. S. Vygotsky, Volume 1: Problems of General Psychology (ed. R. W. Rieber & A. S. Carton).* NY: Plenum Press.

Wertsch, J. V. and Kanner, B. G. (1992). A sociocultural approach to intellectual development. In R. J. Sternberg and C. A. Berg (Eds.), *Intellectual Development* (pp. 328-349). NY: Cambridge University Press.

Whitehead, A. N. (1929). *The Aims of Education*. New York: Macmillan.