Situated Sense-Making: A Study of Conceptual Change in Activity-Based Learning

Yam San Chee
Jee Teck Tan
Taizan Chan
Department of Information Systems and Computer Science
National University of Singapore
Lower Kent Ridge Road
Singapore 0511
Email: cheeys@iscs.nus.sg

Abstract

Sense-making is an essential process in learning for understanding. We describe a study of sense-making involving two pairs of students learning basic elements of the visual programming language Prograph. The study emphasizes the critical role of activity in mediating concept development and refinement. Video protocols of learning behavior were recorded and analyzed. The analysis focuses on the situated nature of the meaning construction process. It reveals how exploration, explanation, and expectation play important roles in the sense-making process.

Introduction

In everyday learning as well as in situations of formal education and training, sense-making is an important cognitive activity. In recent years, researchers such as Carroll (1990), Clancey (1991; Clancey & Roschelle, 1991), and Roschelle (1992) have engaged in influential attempts to probe the nature of such activity. In the context of these efforts, this paper reports on an empirical study of two pairs of students learning basic elements of the visual programming language Prograph™.

In the sections that follow, we begin by setting out the theoretical underpinnings of this research. Next, we describe the study approach and the study setup. Protocol segments are then presented and analyzed. We conclude by summarizing what has been learnt.

Theoretical background

This research is theoretically rooted in the writings of Vygotsky (1978, 1986, 1987). An important idea of Vygotsky's thought was that activity, especially in a socially-laden setting, could be fruitfully used as an explanatory principle in psychological study. The stimulus–response (S–R) paradigm can thus be modified by letting activity take the place of the hyphen in the S–R formula to yield *object* ↔ *activity* ↔ *subject* (Kozulin, 1986). In this way, activity is given an important role as the factor that mediates conceptual change through a learner's interaction with external objects. Consequently, we view interaction with some concrete system capable of behavior (acting and reacting to a learner's actions or inputs) as essential to successful sense-making.

Our attention has also been drawn to the relevance of research by Chi and her colleagues (eg. Chi, Bassok, Lewis, Reimann, & Glaser, 1989; Chi & VanLehn, 1991) on self-explanations. Chi et al. (1989) describe a self-explanation as a comment about an example statement that contains domain-relevant information over and above what was stated in a worked-out example. They found that the amount that students learned while studying worked-out examples is proportional to the number of self-explanations generated by the students while studying examples. Our case study observations indicate that *explanations* also play an important role in sense-making activity. In the context of our study, however, explanations are made not only to the self, but also to a collaborating student. In contrast to the work of Chi et al., our students were not learning from worked-out examples (that is, textbook-like examples with explanations). Rather, they only had example Prograph programs to inspect and the Prograph environment to navigate within.

Our empirical observations lead us to propose that *exploration* and *expectation* also play important roles in sense-making. When students face an entirely new situation (the Prograph programming environment in our study), they rely on fairly *ad hoc* exploration to bootstrap the sense-making process. In addition, expectations based on prior knowledge quickly begin to actively shape the sense-making process. Our protocols suggest that expectation-driven learning is a prominent feature of sense-making in general. More successful learners explicitly create new expectations of system behavior, then proceed to test whether the expectations hold in practice—a form of hypothesis testing.

In the description of the study that follows, we provide evidence for the phenomena of exploration, explanation, and expectation in learning activity for sense-making.

Study approach and setup

The method adopted for this study is that of video protocol analysis. It is similar to that employed in Clancey & Roschelle (1991) and Roschelle (1992). The analysis focuses on verbal as well as gestural protocols. It includes some element of conversation analysis (Goodwin & Heritage, 1990; Schegloff, 1991).

Programming language and learning environment

This study is based on the learning of basic elements of the Prograph™ programming language. Prograph is a language developed by The Gunakara Sun Systems for the Macintosh computer. It is a high-level, pictorial, object-oriented programming environment. The focus of our study has been restricted to the dataflow and notational aspects of the language.

Subjects

Our subjects were two pairs of first-year computer science undergradautes. They had completed a semester of Pascal and Cobol programming. However, they had never before encountered the notions of dataflow languages or visual programming. The first pair of students comprised Luke (L) and Charles (C); the second pair comprised Jason (J) and Benny (B) (all names are fictitious).

Task

Each pair of students was provided with a set of basic instructions for valid manipulations in the Prograph interface (eg. how to add new nodes and links, how to delete nodes and links, how to double-click on a node to get more information on that node, etc.). The students were then asked to study six already-coded Prograph programs (*methods* in Prograph parlance) of increasing complexity. They were informed that they would later be required to code a subprogram to an existing program. They were, therefore, to study the six programs with a view to completing that task at the end. Students were given no information on the meanings of the (visual) language notations. They were expected to interact with the programming environment to make sense of the given programs.

Presentation and analysis of protocols

In this section, we analyze two protocols to illustrate the features of exploration, explanation, and expectation in the sense-making process. We do not attempt to examine these features one at a time because they usually co-occur within episodes.

Protocol 1

In this protocol excerpt, we examine how Luke and Charles try to make sense of program *Third* (see Figure 1).

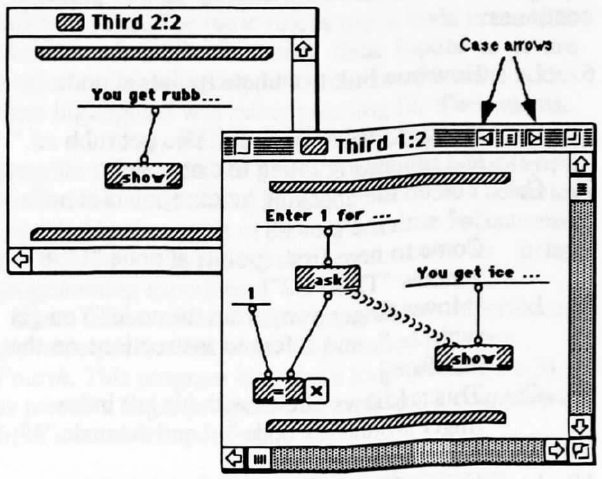

Figure 1. The program *Third.*

When program *Third* executes, a dialog pops up telling the user, "Enter 1 for ice cream, others for rubbish." (see window "Third 1:2"). The program then checks whether the user's input is a "1". If it is, it shows the user a dialog saying, "You get ice cream!".[1] If the input is not 1 (the **x** symbol next to the operation node denotes *not*), control passes to the code in the next case (see window "Third 2:2") and a

[1] The link between the operation icons "ask" and "show" is called a *syncro*. It specifies that the operation "ask" should execute before the operation "show".

dialog stating "You get rubbish!" appears. The labeled buttons are case arrow buttons that allow the user to switch the active window from one case to the other.

A protocol excerpt between Luke and Charles relating to the above program follows. Notations employed in the protocols are explained in the Appendix.

1. L: How did this [points at the node "=" in window "Third 1:2"]
   come here? [points at the node "You get rubb . . ." in window "Third 2:2"]
2. C: With some link . . .
   Down here::: [points to the case arrow buttons]
3. L: No, that's the programming, but the programming . . .
4. C: The what =
5. L: = the programming link.
   This [points at window "Third 1:2" and then to window "Third 2:2"]
   is just input text.
   Go to control. [points at "Controls" menu item]

After two additional exchanges, the protocol continues:

6. L: How you link from here [points at node "="]
   to here? [points at node "You get rubb . . ." and remains pointing to that node]
7. C: You do the checking here::: [points at node "=" with the mouse]
   Come to here first. [points at node "1" in window "Third 1:2"]
8. L: [Moves finger away from the node "You get rubb . . .", and refers to instructions on the interface.]
9. C: This::: [draws circles with his left index finger around the node "=" and the node "1"]
   The checking part . . .
10. L: Hm::: You go here . . . [points at node "="]
    You delete the link, see what happens? [still pointing at node "="]
11. C: What do you mean?
12. L: The link from here [points at node "="]
    to here::: [points at node "You get rubb . . ."]

In the above excerpts, the students were puzzling over how the flow of program control passed from the window "Third 1:2" (henceforth referred to as "Window1") to the window "Third 2:2" (henceforth referred to as "Window2") when the input was not equal to one. (This concern with control flow is not surprising in light of prior experience with procedural programming languages only.) Having become accustomed to the way in which program execution moves from node to node and how the nodes are connected by links, they formed the *expectation* (through generalization) that there must also be a link between Window1 and Window2. Consequently, they expended considerable time and effort, including exploratory effort, trying to find it. (In actual fact, there is no such link.)

The protocols reveal that the students' concept of a link at this point in time was still hazy. Consequently, they failed to "connect" while conversing with one another. While Luke was talking about program control flow (and confusing physical data links with the idea of flow of control), Charles was responding with how Window2 could be obtained from Window1 by clicking on the right-pointing case arrow (something he had discovered by accident earlier).

It is unclear what Luke meant by the statement "This is just input text." in segment 5. Possibly, he was suggesting that the code in Window2 simply allows the programmer to "input" the appropriate text for display.

To get out of the understanding impasse, Luke suggested exploring the "Controls" menu item. This attempt at *exploration* appears to be entirely *ad hoc*. Inspection of the items on the menu did not provide any useful information, and the students reverted to their discussion of the hypothesized link.

Segment 10 of the protocol illustrates a further instance of *exploration*. Being unable to form a coherent understanding of how to establish the "missing link" between Window1 and Window2, the students decided to engage in further exploration ("see what happens"). This time, Luke suggested deleting the missing link. Charles tried to use the same procedure for deleting a link between two icons within a window but encountered difficulty because he was trying to delete a nonexistent link *between* windows. As this attempted exploration was based on confused thinking and the absence of any clear expectation of what would happen, it is not surprising that the attempt yielded nothing useful in terms of conceptual understanding. This excerpt illustrates how exploration in the absence of expectation can lead to failure in sense-making.

Protocol 2

In this protocol, we explore the features of expectation and explanation occurring in a highly situated learning context. The two students here, Jason and Benny, were puzzled about the order in which the different nodes were highlighted when they traced through the execution of program *Second* using the environment's code tracing facility (see Figure 2).

When this program is executed, a dialog box pops up saying, "Enter something". The dialog box contains a user response area that contains the default response "default". After the user types in some input, the show operation on the left-hand side executes and the message "First Hello: default" (or whatever the user typed in for "default") appears, followed by a second message "Second Hello: default". The syncro link dictates that First Hello will execute before Second Hello.

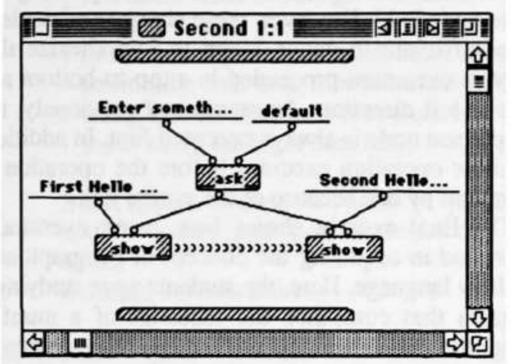

Figure 2. The program *Second.*

A protocol excerpt related to the students' study of program *Second* follows:

1. J: [Traces through the program from the beginning, then retraces program.] Hm::: ... What's going on?
2. B: You don't (enter) ... You don't enter, (it's) like anything will be default.
3. J: OK. [hits return; node "First Hello ..." is highlighted] First hello.
4. B: Yah.
5. J: How come some of the lines is? ... Don't think so ... [may have noticed that the color of links from executed nodes changes from green to blue] First hello [hits return; node "default" is highlighted] Default [hits return; node "Enter someth ..." is highlighted] enter something [hits return; node "ask" is highlighted; hits return; prompter window is displayed; types in characters; hits return; prompter window is closed; node "show" is highlighted] Show ...
6. B: Press (enter) ...
7. J: [Hits return; dialog window is displayed.]
8. B: That's what you entered ...
9. J: First hello. [points at the centre of the dialog window and slides finger to the top left corner of the screen]
10. B: Because that's what you entered.
11. J: [Hits return; dialog window is closed; node "Second Hello ..." is highlighted.] Show there::: [points to the node "show" on the left] First hello::: [points to node "First Hello ..."]
12. B: Yah ...
13. J: Then what's the next one? Second hello. [points at node "Second Hello ...", then points to node "show" on the right]
14. B: Yah, second hello.
15. J: So is ... [hits return; node "show" is highlighted] second hello [points to the node "Second Hello ..."] will be showing also [hits return; dialog window is displayed; hits return; window is closed]
16. B: Never ask you again?

The above excerpt illustrates how the students tried, without much success, to make sense of the order in which program execution proceeded (as reflected by the order in which different nodes become highlighted). Unfortunately, the order in which Prograph selects nodes for execution is largely indeterminate. The basic rule is that a node is ready to execute when all necessary data inputs to it are available. Consequently, the order in which the nodes were highlighted was rather puzzling for the students.

In addition, Benny expressed surprise that the program did not ask for a new input before displaying the Second Hello dialog (segment 16). This behavior is probably the result of having become accustomed to paired input–output sequences in prior programming experience.

Some fifteen minutes later, the students experienced the same puzzlement when they studied program *Fourth.* This program includes a loop (see Figure 3) as part of a larger program that performs a countdown from a number input by the user.

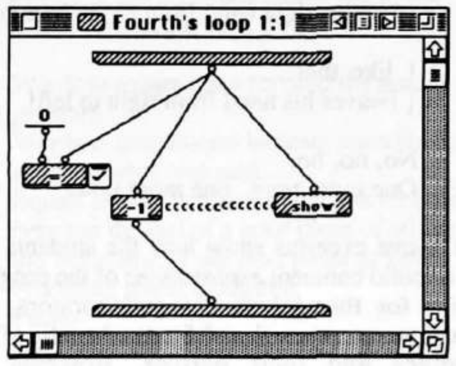

Figure 3. The loop code of program *Fourth.*

In the execution of the program loop, the "=" operation node is always executed first because it involves a comparison operation. The input to the loop is compared with the number zero. If the number is zero, execution terminates (indicated by the tick with a bar above it); if not, the node "show" is executed, followed by the node "–1" which decrements the current number entering the loop by one.

The protocol excerpt follows:

17. J: { How come its lateral?
{ [slides finger from node "show" to node "- 1"]

{ How come it's moving sideways?
{ [moves left hand from right to left]

{ I think the order is like that
{ [waves hand from top to bottom of the screen]

{ from top down:::
{ [waves hand from top to bottom of the screen]

{ from top going down
{ [points finger at the input of the loop, slides finger down to node "="]

{ That's why you . . .
{ [points finger at the input of the loop, slides finger down to node "="]

{ you have to go to this thing first.
{ [points finger at the input of the loop, slides to node "="]

{ I think it is like that
{ [waves right hand from right to left]

{ like that
{ [waves right hand from top to bottom]

{ like that . . .
{ [waves his hand from right to left]

No, no, no.

18. B: One more time, one more time.

The above excerpts show how the students were trying to build coherent *explanations* of the program's behavior for themselves. As collaborators, their explanations are mutual and for the benefit of both themselves and their partner. However, the explanation activity is deeply entwined with building and testing *expectations* in order to determine whether tentative suppositions made are supportable (see segments 2–4, segments 7–12, and segment 17). For example, in segments 7–12, when the dialog window appears, Benny notices (as he expected) that the words typed in by Jason appear in lieu of the word "default". In segment 10, he explains that this outcome is the consequence of what Jason entered ("*Because* that's what you entered.").

Segment 17 illustrates the highly situated nature of sense-making activity. It was purely coincidental that, in the code for Fourth's loop, node "=" was positioned highest, and node "show" was positioned just slightly above node "–1". However, these chance positionings were sufficient to cause Jason to hypothesize that program execution proceeded in a top-to-bottom and right-to-left direction. As mentioned previously, the comparison node is always executed first. In addition, the show operation executes before the operation to decrement by one because of the syncro link.

The final excerpt shows how Jason eventually succeeded in acquiring the concept of Prograph as a dataflow language. Here, the students were studying a program that computes the factorial of a number using iteration. The loop portion of this program is shown in Figure 4. There are three operations involved: a comparison with zero that terminates if true, an operation to decrement an incoming number by one, and an operation to multiply two numbers.

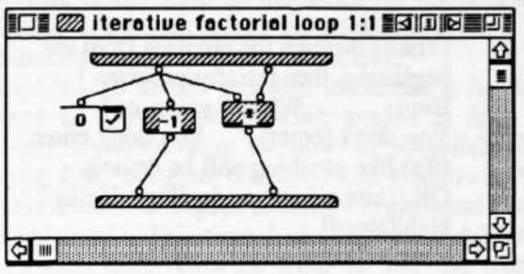

Figure 4. The loop code of program *Iterative Factorial.*

The protocol excerpt follows. In this excerpt, all pointing is carried out with the tip of a pencil. The verbalization was uttered as a statement of expectation *before* the code was traced.

19 J: So, we should have two inputs::: . . .
[points at the two inputs of the loop code]
This [points to node "0"]
is to check whether it's zero right?

{ Then the value will go into three places
{ [slides tip of pencil from left input along the middle data link, then from left input along the left data link, then from left input along the right data link]

{ One place,
{ [slides tip of pencil from left input along

the middle data link]

{ two place,
{ [slides tip of pencil from left input along the left data link]

{ three place . . .
{ [slides tip of pencil from left input along the right data link]

{ The same value should go into three places . . .
{ [slides tip of pencil from left input along the middle data link, then from left input along the left data link, then from left input along the right data link]

But::: [points to node "0", node "-1", and node "*", and repeats pointing sequence again]
it should have an order:::

This protocol segment illustrates how the notion of dataflow was eventually conceived. Jason stated first that "the value will go into three places" and then again that "the same value should go into three places." This explicit articulation represents a significant achievement in conceptual change. However, the old vexing problem concerning the order in which the operations should execute remains unsolved.

Conclusion

Our study of students engaged in collaborative sense-making reveals that the processes of exploration, explanation, and expectation are important features of learning. Students must engage in activity in order to build an understanding of the domain under study. Such an understanding usually requires the construction a causal mental model. Only through interacting with the objects of an external environment can deep learning be accomplished.

References

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

Chi, M. T. H., Bassok, M., Lewis, M. W., Reimann, P., & Glaser, R. 1989. Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science* 13:145–182.

Chi, M. T. H. & VanLehn, K. 1991. The content of physics self-explanations. *The Journal of the Learning Sciences* 1(1):69–105.

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

Clancey, W. J., & Roschelle, J. 1991. Situated cognition: How representations are created and given meaning. Paper presented at the AERA Symposium, *Implications of Cognitive Theories of How the Nervous System Functions for Research and Practice in Education*, April 1991, Chicago.

Goodwin, C., & Heritage, J. 1990. Conversation analysis. *Annual Review of Anthropology* 19:283–307.

Kozulin, A. 1986. The concept of activity in Soviet psychology. *American Psychologist* 41(3):264–274.

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

Schegloff, E. A. 1991. Conversation analysis and socially shared cognition. In L. B. Resnick, J. M. Levine, & S. D. Teasley (Eds.), *Perspectives on Socially Shared Cogntion*. Washington, DC: APA.

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

Vygotsky, L. S. 1986. *Thought and Language* (A. Kozulin, Ed.). Cambridge, MA: MIT Press.

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

Appendix

The meanings of the notations employed in the protocols are explained below.

[ ]	Words in square brackets describe actions and gestures.
( )	Words in parantheses indicate transciber's best guess of what was said.
=	Equals sign indicates that there is no break between the end of a prior piece of talk and the beginning of the next piece of talk.
:::	Preceding sound elongated.
. . .	Pause.
{ {	Overlapping verbal utterance with action or gesture.
?	Question intonation.