A Formative Evaluation of Scenario-Based Tools for Learning Object-Oriented Design

by Hope D. Harley, Cheryl D. Seals, Mary Beth Rosson

Abstract

Advances in computing have awakened a century old teaching philosophy: learner-centered education. This philosophy is founded on the premise that people learn best when engrossed in the topic, participating in activities that motivate learning and help them to synthesize their own understanding. We consider how the **object-oriented design (OOD)** learning tools developed by Rosson and Carroll [5] facilitate active learning of this sort. We observed sixteen students as they worked through a set of user interaction scenarios about a blackjack game. We discuss how the features of these learning tools influenced the students' efforts to learn the basic constructs of OOD.

Introduction

Education is a growing application area for computing technologies. Advances in the power and sophistication of interactive computing have awakened a century old teaching philosophy: learner-centered education. This philosophy is founded on the premise that people learn best when engrossed in the topic, participating in activities that motivate learning and enable them to synthesize their own understanding. In this paper, we examine the learning process entailed in developing an understanding of object-oriented design (OOD). Traditional instruction tends to emphasize the product of the design and development process, but not the process itself [3]. Research has shown, however, that students benefit from explicit instruction in complex design skills [3]. Rosson and Carroll [5] have developed a set of tools for learning OOD from scaffolded examples. They have examined the usefulness of their approach in a tutorial for human-computer interaction (HCI) professionals. This paper examines the efficacy of these tools in a university environment where students are exposed to the OOD paradigm for the first time.

Design and development are usually taught by beginning with simple examples and advancing to more complex problems. Learner-centered education uses a problem-based approach, where the problems force the learner to find and make use of new knowledge and skills in order to solve the problem at hand. The goal is active exploration, construction, and learning as opposed to textbook reading and lecture comprehension. The learning process is oriented towards a set of realistic and intrinsically motivating example problems. A problem-oriented approach such as this is well-suited for the task of learning OOD, where the fundamental goal is to analyze a problem situation and decompose it into an object-based representation [7].

A technique often used for learner-centered education is **scaffolding**: support for learners that enables them to engage in activities that are normally out of their reach. Scaffolding often involves modeling a process for the learner, coaching the learner through that activity, and then providing opportunities for the student to articulate what has been learned. Rosson and Carroll [5] provide scaffolding in their OOD learning exercises. They begin their examples with user interaction scenarios. Each scenario presents a piece of an application design problem (a description of a user pursuing a goal within the system). They claim that scenarios provide scaffolding by breaking a large abstract problem into small concrete subproblems. Furthermore, the scenarios help learners identify candidate objects: they can simply begin with the problem entities mentioned in the narrative.

In Rosson and Carroll's learning environment, students first encounter OOD constructs by exploring pre-analyzed scenarios. Then they refine and extend the set to practice what they have learned. They start with a scenario list for an example problem (upper left of Figure 1). In another window, the Point of View (POV) Browser (upper right of Figure 1) presents an analysis of a selected scenario from the perspective of individual design objects. A POV for a particular object details its role in the scenario, the information it manages (its parts), other objects it works with (its collaborators), and the behaviors it provides (its responsibilities). As scaffolding, POVs help learners grasp the important OOD concepts of encapsulation and collaboration. They also encourage learners to anthropomorphize problem entities, an important technique in responsibility-driven OOD [7].

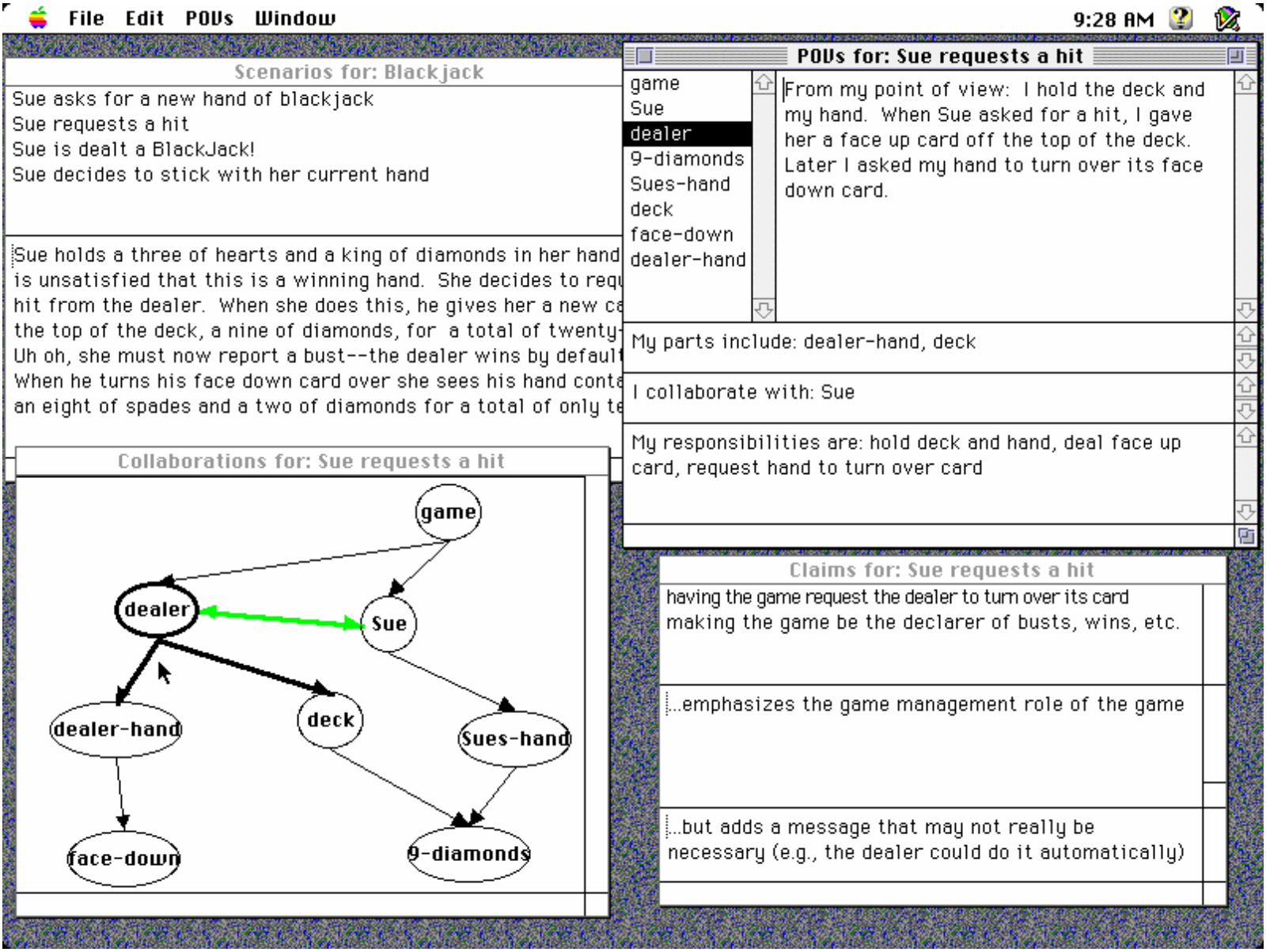

Figure 1: Scenario Browser (upper left), POV Browser (upper right), Collaboration Graph (lower left) and Claims Browser (lower right).

Examining object POVs in a scenario leads the learner to another window containing a Collaboration Graph (lower left of Figure 1). This graph displays all of the objects in a scenario as well as the relationships between them (i.e., as parts and collaborators). As an overview of the entire object model for a scenario, a collaboration graph also provides learners with an opportunity for reflection. As in any design domain, the design process is iterative. Reflecting on the relationships among objects as they are represented by the graph causes the learner to identify new objects as parts. Collaborators are refined and new responsibilities identified as the object interaction narratives are elaborated. Revisions made to the POV browser update the Collaboration Graph accordingly. As design decisions are made, the Claims Browser (lower right in Figure 1) can be used to document hypothesized positive (middle pane) and negative (bottom pane) consequences of individual design features.

In our study, we considered how a scaffolded example presented by these tools facilitates the OOD learning process. We observed sixteen students, some working alone and others in pairs, as they examined and extended a set of blackjack game scenarios. From these observations, we tried to understand how features of the OOD tools influenced the learning of the students; we focused particularly on the role of the POV and collaboration analyses as means of teaching the basic constructions of responsibility-driven design [7]. Although we were primarily interested in the usefulness of the tools, we also examined their usability.

Method

Subjects

The subjects in this experiment were sixteen novice OOD developers, all volunteers. They were near completion of an introductory C programming course at Virginia Tech and most were freshmen planning to pursue degrees in computer science. Two were in other majors (math and mechanical engineering), taking the course to fulfill an elective requirement. All were familiar with classical procedural languages such as C and Pascal. A few had some experience with C++, but no formal training in OOD.

Materials

The students were given printed instructions to guide the exploration process; each tool was introduced with a single ``guided exploration card''. During the learning experience, subjects explored three documented blackjack scenarios representing typical game-playing episodes: getting a new hand to play, requesting and receiving a hit, and being dealt a blackjack. The object model for each of these scenarios was documented via 8-10 POVs and 2-4 software design claims. After exploring these scenarios, students then elaborated a fourth scenario by defining a set of POVs. This scenario described a player choosing to ``stick'' with their current hand. Instructions for adding these POV entries were given in the printed materials.

Procedure

Subjects worked in pairs and individually to explore the learning materials; six sessions consisted of pairs, and four sessions were for individuals. Not surprisingly, the learning sessions of the pairs were more interesting than those of students working alone. As other researchers have shown [2], collaboration facilitates both successful performance and reflection for learning:

- Groups can solve more interesting and complex problems than an individual working alone;
- Students working in groups need to articulate designs, critiques, and arguments to other group members, encouraging the kind of reflection that leads to learning.

At the beginning of the session, subjects were introduced to the goals of the experiment and completed a pre-test questionnaire aimed at documenting their experience with programming and with object-oriented languages in particular. They were then invited to play a few hands of blackjack; this was done to help them focus on the underlying game model, and to ensure that all learners were familiar with the game of blackjack. Then, as in Rosson and Carroll [5], subjects began their OOD learning experience by exploring pre-analyzed scenarios as an orientation to the OOD concepts of encapsulation and collaboration. After examining the documented scenarios and the associated design representations presented in the POV Browser, Collaboration Graph, and Claims Browser, the learners were prompted to contribute to the design by adding POVs for the final (incomplete) scenario in the browser. After completing their work on this scenario, subjects completed a post-test to assess their reactions to the learning experience.

Subjects were videotaped and their interaction with the system recorded by screen capture equipment for later analysis. An experimenter intervened only when asked a question by the subjects or when subjects appeared confused about the next step of their task.

Results

Overview

All subjects completed at least a rudimentary solution to the problem. Generally, the solutions contained the major objects needed by the fourth scenario (e.g., game, dealer, player, cards), along with a basic description of each object and its corresponding parts, collaborators, and responsibilities. The amount of time spent in designing a solution averaged 35-40 minutes, with three groups spending more than 50 minutes to complete the task. Some solutions included as few as eight objects while others had as many as eleven. The minimum number of part or collaborator relationships in the collaboration graphs was one while the maximum was fifteen. In the following sections we present a detailed analysis

of two groups, hereafter referred to as A and B. These groups were chosen for detailed analysis because of the method in which they interacted with the materials and the careful and interesting reflection that occurred during their episodes.

Detailed Analysis: Group A

Group A consisted of two freshman students (referred to as P1 and P2) who were both comfortable with programming. P1 was not familiar with object-oriented concepts, while P2 reported a basic understanding of object-orientation. The group spent 8 minutes interacting with the blackjack game and studying the first three scenarios.

As these two students began their investigation of the POVs for the three scenarios, P2 almost immediately made a connection between the objects analyzed in the scenario and the real world situation (see Figure 2): while reading the narrative description about the deck object, he asked ``Would there be fifty-two objects in a real deck?'' The narrative for the deck's point-of-view stated that five cards were dealt to the players of the game. P2's abstraction to a real world deck showed that he had a basic understanding of the relationships we were trying to establish. P2 also stated that messages sent to the objects affected how the objects would react. While looking at the collaboration graph, Group A inspected the object relationships. At this point, P2 again indicated his level of understanding about interactions among objects, noting that the arrows connecting the objects depicted which way messages were passed.

The pair then proceeded to the Claims Browser. Here, P2 realized that the Claims Browser contained the design rationale. He read Claim 2: ``Giving the game management job to the game object''. From this he surmised that this was similar to the dealer object managing the deck. We believe that the concreteness of the scenarios and of the object POVs encourages this sort of reasoning. After reading a claim regarding the impact of ``Letting the deck shuffle itself'', he commented that this was not a real world situation, again suggesting that he was making good contact between his mental model of a blackjack game and the anthropomorphic object-oriented model conveyed by the POVs.

Now Group A was ready to add to the design. Initially, they had problems adding POVs. Although they had been given printed instructions for adding POVs, they seldom consulted them. So, for example, they needed help understanding that the text field prompt for specifying an object's part could be used to specify just one part at a time.

Looking at the Collaboration Graph initiated discourse about message passing and relationships between objects (see Figure 2). The users discussed the objects in the scenario and their methods. P2 expressed his understanding that messages operate on objects.

When starting to add new POVs, P1 and P2 chose to use as an example the POVs defined for another scenario. We had not anticipated the use of POVs in this manner, but students often prefer to use examples as a learning tool [1]. They first specified the list of objects, then began to add the details. They started with the first object, Sue, and added that object's collaborators, responsibilities, and parts, then proceeded down their object list. These students seemed comfortable with the anthromorphic language encouraged by the POV analysis, making comments about what an object ``knows'' and about them ``doing things''. However, they did not fill in the point-of-view narratives for the objects. The process of adding this narrative must have appeared tedious to these students, since they provided very little text for any of the objects. It may also be that their reasoning about parts and collaborators was sufficient to understand the role each object was playing in the scenario.

As shown in Figure 3, Group A identified nine objects: deck, cards, value, face up or down, cards dealt, Sue, Sue's hand, dealer, and dealer's hand. They show the deck as having cards as its part. Of particular interest is their decision to give the card object two parts: a value and a face-up-or-down attribute. These decisions are a true extension to the design, as none of the earlier scenarios modeled these as separate objects (see also Figure 2). In Smalltalk everything is an object, so these students' model is a more accurate depiction of how the scenario would actually be implemented. However, in the example documentation, low-level objects such as numbers and boolean values were omitted for purposes of simplification. It is interesting to speculate that the POV's emphasis on objects and responsibilities led the learners to ``objectify'' even low-level data such as this.

However, while these learners did include low-level attributes as objects, they seemed to have difficulty distinguishing between object instances (e.g., individual cards) and collections of object instances (e.g., the deck, and their objects labeled as ``cards'' and as ``cards dealt''). This may be due to the emphasis of most programming languages on data structures such as arrays rather than on concrete data elements. It may also be that the example POVs of related objects in other scenarios

were not sufficiently clear concerning the complementary responsibilities of card-containers (decks, hands) and the cards themselves.

Another interesting characteristic of Group A's solution is the absence of a game object. Note that this object (which was included in all other scenarios) is an abstraction analyzed for software design purposes: it modularizes the game-management functionality, as well as serving as a ``container'' for the dealer and player objects. However, in the real world there is no blackjack game object; game management is the responsibility of the dealer. We noted earlier P2's connection between the POVs and the real world model of blackjack -- it may be that this strong connection to real world objects decreased these learners' attention to the higher-level blackjack game abstraction. The two-way collaboration defined between the dealer and Sue reinforce the interpretation that these learners were using the dealer to manage game-playing activities.

The post-test responses of P1 and P2 suggest that both felt their experience made them more aware of how the different parts of an object-oriented design collaborate. P1 stated that the Collaboration Graph in particular helped in visualizing object interactions. P2 stated that the learning experience was interesting, but added nothing new to his object-oriented knowledge base.

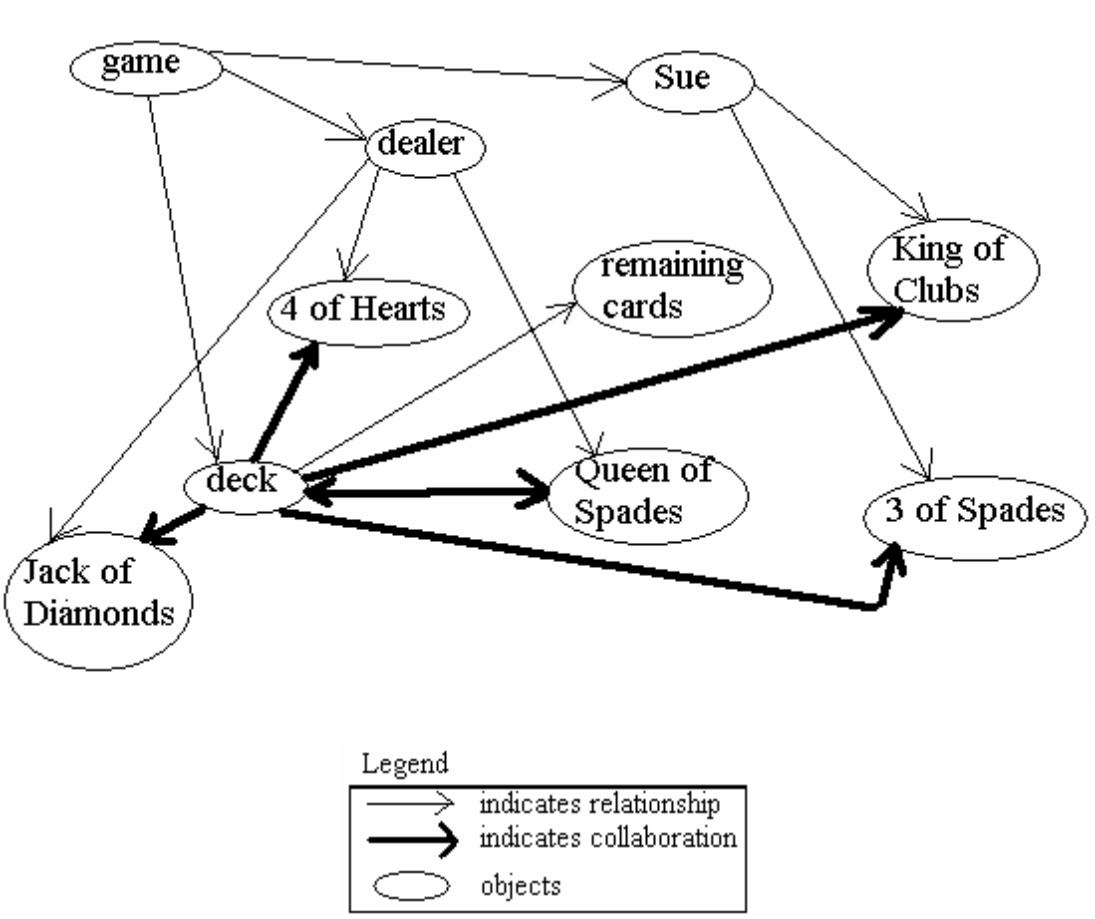

Figure 2: Group A Activity Graph

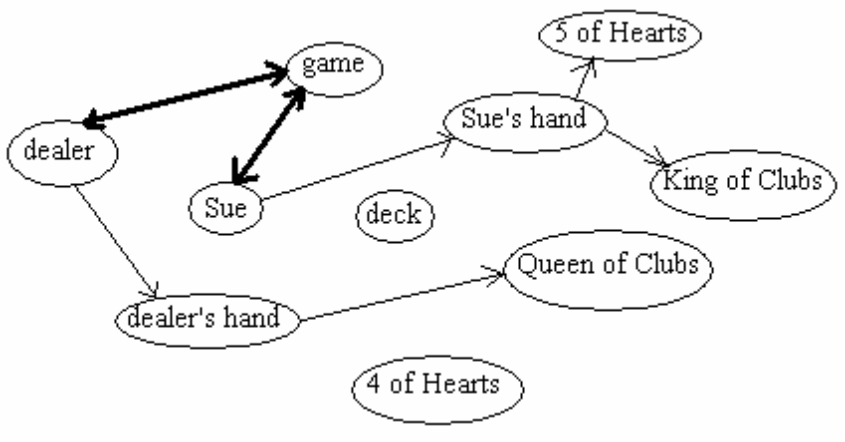

Figure 3: Group A Collaboration Graph

Detailed Analysis: Group B

Group B consisted of two freshmen computer science students who were both comfortable with programming, but neither participant was familiar with OOD. They played the blackjack game and studied the first three scenarios for 14 minutes. After completing these preliminaries, they explored the POVs for these scenarios. Both participants read through all the objects and their descriptions. Next, following their printed instructions, they explored the remaining tools with only brief explanations from the experimenters.

After studying the existing blackjack design, Group B began the exercise. They read the directions and opened the POV Browser. They then went back and re-read the fourth scenario. This review prompted P3 to propose immediately many of the objected needed by the fourth scenario: "We have Sue, the deck, and the dealer and the game itself". This immediate and direct identification of candidate objects from the scenario text suggests that the concrete situation described in the scenarios does indeed aid this first but crucial step of OOD.

P3 suggested that they begin with the game object. They added this object, but instead of elaborating this object's POV, P3 stopped to reflect on his reason for choosing the game first: he explained that he was thinking from the top down, then recanted and said that it might be better to start from the bottom up by beginning with the cards. This is an interesting reversal, as a scenario-based approach to OOD is inherently instance-centered and thus bottom-up: full-fledged abstractions are realized by generalizing across instances analyzed in specific situations [4].

Group B then proceeded to add entries for the cards. Interestingly, they considered but rejected the idea to model the two card hands as separate objects. Perhaps, as we speculated regarding Group A's omission of the game object, calling out a special object to serve as a hand is too much of a departure from the real world situation. Their decision not to model the hand is particularly interesting given their language when discussing card behavior, where they freely refer to "Sue's hand" and the "dealer's hand", but seem to see those responsibilities as subsumed by the two player objects. As we suggested earlier, it may be that refining the POV narratives for the card hands in the example scenarios would help learners see them as independent objects with their own responsibilities. (It may also have been that P3 and P4 were trying to minimize their specification work; merging the hand and player objects does simplify the solution.)

This group's discussion of card responsibilities revealed an understanding of real-world-to-virtual-world mappings: the students were able to see that each card was a persistent object, with different roles at different points of the scenario. Thus P3 proposes that the cards initially are a part of the deck and later became a part of a hand. P4 agrees with this explanation. Another interesting feature of their analysis of cards is their addition of a 'remaining cards' object. They seem to view this object as a placeholder for all the other cards not mentioned in this scenario. This may reflect an effort to connect this scenario with the larger design; it may also be an attempt to depict a more complete model of the deck object (recall Group A's comment about a card deck having 52 cards).

Within 30 minutes, Group B had added all of the objects they deemed necessary, and their associated collaborators, parts and responsibilities. We then suggested that they review the collaboration graph generated by their design. This high-level view of their design alerted them that they had incorrectly added two POVs: they had specified two objects on a single interaction and the system had represented this as one object. They made the necessary correction and were

satisfied with their design. (Recall that Group A had a similar usability problem with the entry field used for specifying object names.)

Like Group A, this group paid little attention to building POV narratives for each of their objects. Although they made no specific comments about this piece of the POVs, we again suspect that they viewed them as tedious and unnecessary.

In their post-test, Group B stated they still had little over-all knowledge of the object-oriented paradigm. However P3 stated that he felt he learned the basis of object-orientation by better realizing the relationships between objects and how those objects interacted with each other, and P4 stated that he gained a better understanding of how objects collaborate and that objects could be reused. Even though Group B felt they had only marginal gains, their resulting design (see Figure 4) shows that they did gain a rudimentary understanding of object-oriented concepts (i.e. object parts, collaborations and responsibilities).

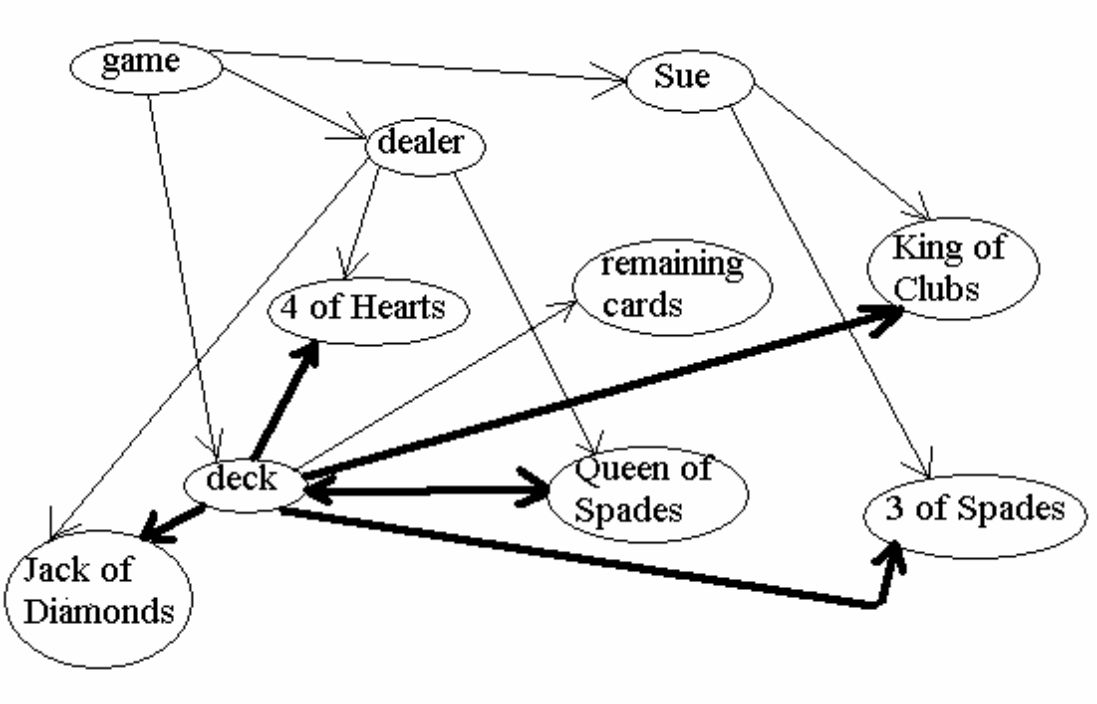

Figure 4: Group B Collaboration Graph

Detailed Analysis: Group C

Group C was made up of one student, a mechanical engineering major who was considering changing his major to computer science. He reported being comfortable with programming but having only a basic understanding of object-oriented concepts. Perhaps because he was older (a senior) and working alone, this subject seemed to take the experiment very seriously, studied all the information available carefully, and often asked questions to better understand the materials. However he rarely provided any comments while he was working.

P3 played blackjack and explored the existing design for about eight minutes. When the experimenters prompted him to begin his exercise, he reread the scenario and opened the POV Browser. He also reviewed the other scenarios, carefully studying the objects defined for them. He then asked for clarification of the POV concept. An experimenter reviewed the basic intuition behind a POV -- different objects in the scenario and their respective views within the scenario. P3 again reread the scenario and added his first POV, Sue's hand. He then wondered what was meant by ``part''. An experimenter again provided clarification, instructing P3 to think about what would be in Sue's hand. With this guidance, P3 went on to add the cards as parts and also added their responsibilities. He asked if Sue's hand should collaborate with the dealer. An experimenter pointed out that it might be more logical to make Sue's hand a ``part of'' Sue, and then let Sue collaborate with the dealer.

P3 again referred to one of the example scenarios, suggesting that he was working by example to a great extent. He added his second object, Sue, and wrote a detailed narrative of her role in this scenario. He also added parts, collaborators and responsibilities. He continued adding objects in a depth-first fashion, elaborating each one in turn, basically working his way sequentially through the scenario narrative.

P3 inquired about overall collaboration in the game. He wanted to know if the dealer and Sue had to collaborate with the game. The evaluator explained that the game object maintained the rules of the game, and from that perspective, the game should collaborate with the dealer and Sue. P3 then looked at the Collaboration Graph and observed that there was currently no such relationship. He proceeded to add two-way collaboration between the game and both Sue and the dealer.

As we remarked earlier, this participant appeared to be more reflective and concerned with detail than the students in Group A. He proceeded through the exercise in a very systematic fashion, asking questions of the experimenters at key points (e. g., clarifications of POVs and of collaboration relationships). His POV specifications were complete (including a narrative for each object) and were quite consistent with the first three scenarios (e.g., he included hands for Sue and the dealer as well as a game object). The few relationships that are missing (e.g., one of the dealer's cards is not connected to his hand) are not critical to the scenario and we suspect were not addressed due to lack of time. We believe that P3's general success may be at least partially attributed to his maturity level.

On the post questionnaire, P3 stated that his experience enhanced his understanding of object-orientation. He felt that he was exposed to the underlying details of object-oriented designs. He also expressed an interest in viewing the actual code written for the blackjack game, suggesting that he was motivated to dig even more deeply into the design.

Summary and Conclusions

During this experiment, we examined the learning process entailed in developing an understanding of object-oriented design (OOD). Our hypothesis was that using a set of tools for learning OOD with scaffolded examples would be of benefit within the university environment where students are initially exposed to the OOD paradigm.

We are encouraged by the results of our study. All of the groups produced coherent design solutions that showed at least a rudimentary understanding of the OOD paradigm. The participants enjoyed playing blackjack as the start of their learning experience. It helped them to quickly focus on the domain model they were attempting to extend. They all perused the existing design very quickly, initially unaware that they would be required to perform the same type of task. Once the students were aware of the design extensions to be done, they tended to review portions of the existing design and use existing scenario POVs as a template to guide their design additions.

Our initial analyses of these three sessions have suggested several improvements we might make to the tools and the blackjack scenario documentation. For example, a common problem with the POV Browser was the input field used to specify POVs, parts or collaborators; it appears as an open-ended field, and users assumed they might be able to type multiple names at once. Clearly labeling the dialog box or adding a direct manipulation interface supporting object definition and connection within the Collaboration Graph might address this problem. Indeed, given the apparent usefulness of the graph in reviewing and checking the POV specification work, it should be used as the primary mode of defining and connecting objects. Removing the POV Browser would decrease attention devoted to the object-specific scenario narratives, though our observations suggest that only the most attentive of learners are willing to develop these anyway.

With respect to the design documentation, it appears that these learners easily grasped the relationship between the object models and the real world situation. At the same time, reliance on their real world understanding of blackjack may have introduced problems. This is evidenced by discouraging the modeling of objects that do not appear as concrete independent entities in the real world (e.g., the game itself). More careful wording of the POV narratives might emphasize the role of these ``virtual world'' entities. We also observed general problems in analyzing and defining collaboration (versus part) relationships. The ``part of'' relation is a concrete one: an object is part of another object if it is physically contained, or in some other way under the control of the parent object. Collaboration however is much more abstract: objects collaborate when they have related responsibilities and need to make requests of one another. Not surprisingly, the specification of collaboration among objects was much more variable than of part relationships, suggesting that we need to devote more attention to the rationale associated with collaboration.

The analyses presented here emphasize the effects of maturity on learning and doing design activities: Groups A and B,

both freshman groups, spent very little time documenting their solutions. Their level of immaturity often lead to usability problems that could have been avoided by a careful reading of the guided exploration cards that were provided. Generally, the subjects did not consult the printed instructions they were given. In future studies, we will explore the use of alternative means of instruction, such as a hypertext glossary of key terms or audio cues, to guide the learners in their exploration.

Our study verified that having students work in pairs enhances the learning process. Generally, a group will have learners of varying abilities. Explaining design decisions reinforces the abilities of the stronger student, while also presenting an alternative argument that may bridge the cognitive gap of the student struggling to understand.

From a qualitative standpoint, we are quite pleased with the success of these OOD learning tools. After only a few minutes of exploration, most of the students were able to produce reasonable models of the exercise scenario. The students' ability to produce these (admittedly small) design solutions with so little preliminary instruction supports the claim that the exploration of these scenario-based tools can teach the fundamental concepts of OOD. Rosson and Carroll [6] have noted that teaching and learning software development skills is a major and open-ended technical challenge. As technology continues to evolve, we expect that the demand for education and training will continue to increase. In future revisions of our OOD tools, we hope to extend the learning materials to include a more in-depth treatment of the subject matter.

References

1

Chi, M.T.H., and Bassok, M. Learning From Examples Via Self-Explanations. In L.B. Resnick (Ed.), *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser.* Erlbaum, Hillsdale, NJ, 1989, pp. 251-281.

2

Guzdial, M., Kolodner, J., Hmelo, C, Narayanan, H., Carlson, D., Rappin, N., Hubscher, R., Turns, J., & Newstetter, W. Computer Support for Learning through Complex Problem Solving. *Communications of the ACM*, 39, 4 (April 1996), pp.43-45.

3

Linn, M.C., Clancy, M.J. The Case for Case Studies of Programming Problems. *Communications of the ACM*, 35, 3 (March 1992), pp. 121-132.

4

Rosson, M.B., & Carroll, J.M. Integrating task and software development in object-oriented applications. In M.B. Rosson & J.Nielsen (Eds.), *Proceedings of Human Factors in Computing Systems, CHI '95 Conference*, ACM Press, New York, 1995, pp.377-384.

5

Rosson, M.B., Carroll, J.M. Scaffolded Examples for Learning Object-Oriented Design. *Communications of the ACM*, 39, 4, (April 1996) pp.46-47.

6

Rosson, M.B., Carroll, J.M. 1997. Expertise and Instruction in Software Development. In M. Helander (Ed.), *Handbook of Human-Computer Interaction.*

7

Wirfs-Brock, R., Wilkerson, B., & Wiener, L. *Designing Object-Oriented Software*. Prentice Hall, Englewood Cliffs, NJ, 1990.

Hope D. Harley is a graduate student in the Department of Computer Science at Virginia Polytechnic Institute and State University. Her research interests include OOA&D and HCI. In her spare time, she enjoys reading and spending time with her family.

Cheryl D. Seals is a graduate student in the Department of Computer Science at Virginia Polytechnic Institute and

State University. She has worked at IBM in application integration and at Bellcore as a database administrator. Her current interests include visual programming languages, educational software, agent systems, and OOA&D.

Mary Beth Rosson is an Associate Professor in the Department of Computer Science at Virginia Polytechnic Institute and State University. She is the author of *Instructor's Guide to Object-Oriented Analysis and Design with Applications*, has developed and co-taught a number of professional short courses in the Task-Artifact Framework and in object-oriented design, and has authored numerous journal articles, conference papers, and book chapters. She is currently serving as co-principal investigator on over $1 million in funded research studying problems in networked education, distance learning and scenario-based design.