# Pass the Ball:
# Game-based Learning of Software Design

Guillermo Jiménez-Díaz, Mercedes Gómez-Albarrán, Pedro A. González-Calero

Dept. de Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid
C/ Prof. Jose Garcia Santesmases s/n. 28040. Madrid, Spain
gjimenez@fdi.ucm.es, {albarran,pedro}@sip.ucm.es

**Abstract.** Based on our experience using active learning methods to teach object-oriented software design we propose a game-based approach to take the classroom experience into a virtual environment.
The different pedagogical approaches that our active method supports, have motivated us to tailor an architecture that supports the creation of different variations of role-play environments, ranging from open-ended trial and error approaches to highly constrained settings where students can not get very far from the solution. We also describe a prototype that instantiates this architecture called ViRPlay3D2.

**Key words:** Game-based learning, object-oriented design, role-play

## 1 Introduction

Object-oriented software design requires a combination of abilities, which can not be easily transferred to the students in lecture sessions. We have tried a more active approach based on role-play sessions to teach software design. This way, the students have the chance to evaluate the consequences of a given design and test their ideas with the other students that intervene in the role-play. Our experience demonstrated its good results [5] and its empirical evaluation has concluded that participating in the play is more effective than just looking at it. Such good results have motivated the work presented in this paper: the construction of a virtual environment for teaching software design through role-play, that intends to maintain, and even reinforce, the benefits of role-play in the classroom. Using ideas from 3D sport games, the interface lets every student play her role while accessing to related information such as the underlying class diagram or previous steps of the use case execution.

Next Section describes the approach used for teaching software design in the classroom, as the starting point for the virtual environment. In Section 3 the features of different learning approaches supported by the teaching approach are presented. Section 4 describes an architecture for role-play virtual environments (RPVEs) that can be created taking into account the different approaches described in previous section. Section 5 details the proposed virtual environment for role-play and software design. Finally, Section 6 concludes the paper.

## 2 An Experience-Based Teaching Approach

During the last years, we have followed an active learning approach to teach object-oriented design. During each session, the instructor actively involves the students in the comprehension and development of a software application. The sessions are supported by two active learning techniques: CRC cards and role-play. A CRC card [1] represents a **C**lass and it contains information about its **R**esponsibilities –what a class knows and what it can do– and its **C**ollaborators –classes that help to carry out a responsibility. CRC cards are combined with role-play activities in responsibility-driven design. The role-play activities [1] are employed to simulate the execution of a use case. Each participant performs the role of an object that intervenes in the simulation. The role-play simulation forces the participants to evaluate the design created using the CRC cards and to verify if they can reach a better solution.

Every session following this active approach runs in three stages:

1. Pre-simulation stage. The instructor selects a design scenario consisting of a case study (the design problem), an initial (and maybe incomplete) solution, and a set of use cases that will be used during the simulation. According to this scenario, the instructor binds each student to a specific role and provides her with the corresponding CRC card.
2. Cycle simulation-modification. Students are responsible for the message passing, modifying CRC cards and deciding when the simulation finishes. During the simulation, the instructor can help the students when a deadlock happens. Students can also communicate among them in order to know about their responsibilities. The instructor registers the simulation by constructing a Role-Play Diagram (RPD): a semi-formal representation of a scenario execution in an object-oriented application that capture objects' state [1].
3. Evaluation. The instructor evaluates the resulting design. If the design is appropriate, the practical session finishes. Otherwise, the instructor discusses with the students the pitfalls found in the design and analyses possible improvements. In this case, the practical session goes back to step 2.

## 3 Variations of the Learning Sessions

The analysis of the learning sessions has promoted the identification of a set of flexible aspects in the session pattern. These aspects allow us to define learning sessions with a common learning methodology and different pedagogical approaches. For each aspect, we have identified its degree of flexibility and how they affect to other aspects. Finally, we have classify them according to the stage that they affect. We have encountered the aspects listed below:

- **Scenario Selection.** This is a pre-simulation aspect that defines how the scenario is selected before starting the simulation.
- **Role assignment.** This pre-simulation aspect concerns how to assign the roles in the scenario to the students and what to do if there are not enough students to simulate the scenario previously selected.

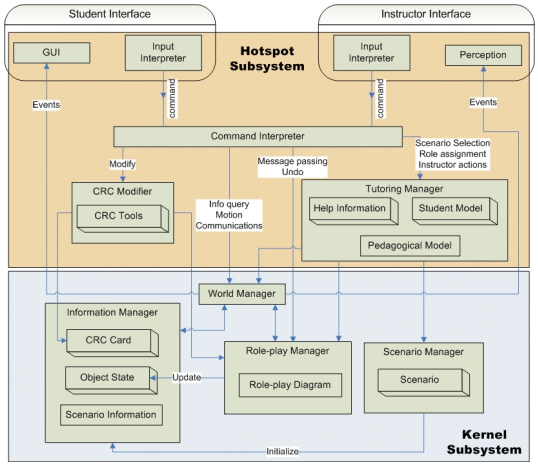


**Fig. 1.** RPVE Architecture

- **Student's degree of freedom.** This simulation-modification aspect determines when the instructor takes the control in order to provide help or to correct the wrong actions of the students.
- **Help provision.** This aspect is related to the previous one and it defines what the instructor does when it takes the control. It is classified as a simulation-modification aspect.
- **Modifications in the initial design.** This is a simulation-modification aspect that defines the modifications the students can do in the CRC cards provided with the scenario. It also concerns the tools and techniques employed to modify the CRC cards, who is responsible for doing the modifications and when they can be done.
- **Instructor.** This is an evaluation aspect that determines how and when the instructor interacts with the students. This aspect is strongly related with most of the previous aspects.

## 4 The RPVE Architecture

We have designed a high level architecture for RPVEs that gathers the variation aspects detailed in Section 3. The architecture is shown in Figure 1 and it clearly differs two subsystems:

- ***Kernel subsystem.*** It implements the behavior that is common to any RPVE. It is formed by several immutable modules and it is responsible for the management of the world entities (*World Manager*); the static information storage, such as the CRC cards (*Information Manager*) and scenario description (*Scenario Manager*); and the update of the dynamic information employed in the simulation, such as RPDs and the object state (*Role-play Manager*).

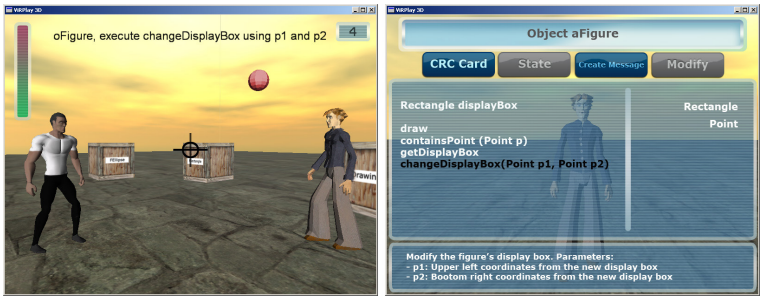


**Fig. 2.** Two screenshots from ViRPlay3D2. On the left, a student's point of view; on the right, the appearance of the object inventory

– ***The Hotspot subsystem.*** It contains the flexible modules within the architecture. Different instantiations of these modules produces different RPVEs. They are related to the communication interface between the users and the RPVE (*Student and Instructor Interface*); the translation of the user input to a set of commands that the rest of the modules understand (*Command Interpreter*); the management of the tools and strategies to modify the CRC cards (*CRC Modifier*); and the management of the tutoring tasks, such as the scenario selection and the role assignment strategies, how the instructor intervenes in the simulation and the help information storage and management (*Tutoring Manager*).

## 5 Transferring the Learning Sessions to a RPVE

Following this architecture, we are developing ViRPlay3D2 (see Figure 2), a multiplayer environment where the students mimic the classroom role-play sessions. The students are immersed using a first-person view that simulates the point of view of the objets that participate in the role-play. An aiming point in the center of the screen serves to interact with other entities in the virtual world and to throw a ball to represent the message passing. The user interface also displays a score that contains information about the current state of the role-play session.

The main elements and actions of a role-play session are represented in the RPVE according to the following metaphor:

**Objects and classes.** Each object is represented by an anthropomorphical avatar controlled by one student. In contrast, a wooden box represents a class. Each box carries a label showing the class name.

**CRC cards and inventory.** Every class and object in the virtual world has an inventory. An object inventory (see Figure 2) contains a CRC card with the object responsibilities and collaborators, and the current object state. A class inventory only displays information about the class constructors.

**Scenario description and role-play diagrams.** The virtual environment contains a desktop that contains the overall description of the played scenario. Moreover, the interface contains a score, which shows the number of role-play steps executed during the current simulation (see Figure 2, on the left). The student can enlarge this score and see the RPD.

**Active object and message passing.** In our environment, the active object is characterized by holding a ball. The execution control is transferred from one object to another by throwing the ball. This throwing represents the message passing and it is divided into three stages:

- Creating the message. When the active object is looking up the inventory of one of its collaborators (or her own inventory), the student can create a message by selecting a responsibility, filling in the actual parameters and clicking on the "Create message" button (see Figure 2 (right)). If an object can send a return message, the inventory contains a special responsibility called "Return". The student is responsible for providing the returned value. A new object is created using the class inventory of one of its collaborator classes.
- Throwing the ball. The avatar that owns the ball throws the ball in the direction that it is aiming. While the avatar throws the ball the environment displays a message with information about the message.
- Catching the ball. This action is available only when the avatar that wants to catch the ball is close to it and the object that represents is the receiver of the message created before throwing the ball. When an avatar catch the ball, it becomes the active object and the RPD and the score are updated with the message details.

**Information retrieval.** The "Look at" action allows the student to see the information related to the entity that her avatar is aiming. The action is executed aiming at an object and clicking with the right mouse button. "Look at" displays the avatar inventory and it contains information about the CRC card. The student can also look at the desktop and display the information about the current simulated scenario. Moreover, if she looks at the ball, detailed information about the current invoked method is displayed. Furthermore, the student looks up her own inventory using the "My Inventory" tool. It displays the inventory of the object represented by the avatar. In this inventory, the student can see the CRC card and the object state. This inventory is also employed to create self-invoked messages.

**CRC modifications, Undo and Communication.** A CRC card Modifier tool is available through the object inventory (see the "Modify" button in Figure 2). Using this tool, a student can add, remove or modify the class responsibilities and collaborators. When the students consider that they have made a mistake when sending a message, the active object can undo this simulation step. When the "Undo" action is performed, the environment updates the state of the objects and the RPD according to the previous simulation step. The ball also returns to the right active object. Finally, the students discuss during the simulation using a chat-like communication tool.

## 6 Conclusions

In this paper we have described the transfer of a successful active learning methodology to teach object-oriented design into a virtual environment. Our own teaching experiences have revealed that this kind of techniques increases student motivation [5] and the students better assimilate concepts in object-oriented design after attending and participating in role-play sessions. CRC cards and role-play are commonly employed in computer science courses [1, 6].

The good results of our experience have promoted to tailor a kind of virtual environments, where the students collaborate to create and evaluate an object oriented design using role-play. Although the use of simulation environments is well-known to teach object-oriented programming [2] we have not found this kind of environments for object-oriented design in the literature.

ViRPlay3D2 is a prototype of an RPVE that builds on our previous experience developing game-based learning environments [3, 4]. After completing its development, we plan is to evaluate its impact in the students learning.

**Acknowledgments.** This work has been supported by the Spanish Committee of Education and Science project TIN2006-15202-C03-03 and it has been partially supported by the Comunidad de Madrid Education Council and Complutense University of Madrid (consolidated research group 910494).

## References

1. J. Börstler. Improving CRC-card role-play with role-play diagrams. In *Companion to the 20th annual ACM SIGPLAN conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 356–364. ACM Press, 2005.
2. W. Dann, S. Cooper, and R. Pausch. *Learning to Program with Alice*. Prentice Hall, Harlow, England, 2005.
3. M. A. Gómez-Martín, P. P. Gómez-Martín, and P. A. González-Calero. Dynamic binding is the name of the game. In *Entertainment Computing - ICEC 2006, Fifth International Conference*, Lecture Notes in Computer Science, pages 229–232. Springer, 2006.
4. G. Jiménez-Díaz, M. Gómez-Albarrán, M. A. Gómez-Martín, and P. A. González-Calero. Software behaviour understanding supported by dynamic visualization and role-play. In *ITiCSE '05: Proc. of the 10th annual SIGCSE Conf. on Innovation and Technology in Computer Science Education*, pages 54–58. ACM Press, 2005.
5. G. Jiménez-Díaz, M. Gómez-Albarrán, and P. A. González-Calero. Before and after: An active and collaborative approach to teach design patterns. In *8th International Symposium on Computers in Education*, volume 1, pages 272–279. Servicio de Imprenta de la Universidad de León, 2006.
6. S. Kim, S. Choi, H. Jang, D. Kwon, Y. Yeum, and W. Lee. Smalltalk card game for learning object-oriented thinking in an evolutionary way. In *Companion to the 21st annual ACM SIGPLAN conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 683–684. ACM Press, 2006.