Incremental Role Play

Jutta Eckstein

Objects in Action, Germany
jeckstein@acm.org
Thierschstr. 20
80538 Munich
Germany

Tell me and I'll forget;
show me and I may remember;
involve me and I'll understand.
Chinese Proverb

THUMBNAIL

The complexity of object-oriented architectures is hard to understand with abstract explanations; therefore ask the students to behave as objects of the design and develop the design in several iterations.

INTENT

How do you teach complex object-oriented architecture without loosing your students?

INDICATIONS

The students already have an understanding of the object-oriented basics, such as sending and receiving messages. The students have to get familiar with a fairly complex architecture, like a framework.

COUNTERINDICATIONS

You don't feel comfortable in performing role plays with this particular audience, e.g. because the audience are members of the board of directors.

The audience size is smaller than 3 students.

The audience size is larger than 20 people. The pattern usable in a larger environment, but the audience will have to be divided into smaller groups.

The course is set up in a distance learning environment.

FORCES

It's much easier for the students to understand a complex architecture when they understand the purpose of the concept; however, we often teach solutions for problems the students never had.

Students understand complex concepts easier if you simplify them, but you cannot simplify some architectures without loosing the idea.

You'd like to provide a positive learning environment, but the difficulties in understanding complex concepts may frustrate the students.

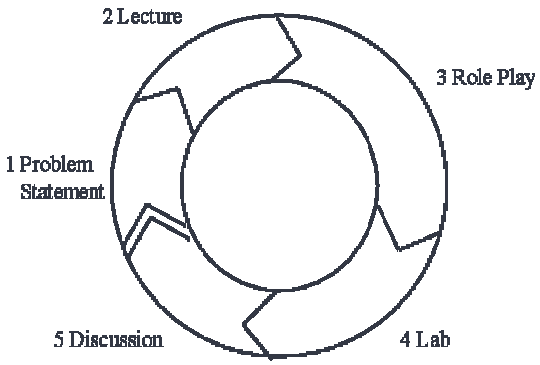
Students understand new material better if they're involved because telling and showing is not the same as involving.

Students often only believe in a technology when they have used it themselves, but it is time consuming to lead them through a whole project.

Students can memorize new stuff better, if you repeat it several times, but it takes time to say the same things over and over again.

SOLUTION

Invite your students to behave as the objects in a role play. Start with a simplified version of the concept and add further aspects as problems show up.



The underlying concept of the pattern is its incremental structure. The architectures are getting more and more complicated over time. The pattern starts introducing a simple architecture, explains a more complex architecture in the next increment and concludes with teaching a large really complicated architecture. The problems discovered in the Discussion phase give raise to the next iteration. It helps if you find several small and medium-sized frameworks from which to teach, so the students are getting a good understanding about layering of architectures.

STRUCTURE

Problem statement: Make the students attentive to the problem the architecture solves.

Lecture: Explain the architecture or rather the concept so the students get an understanding of how the problem is solved.

Role Play: Every student plays one part of the concept to get a deeper knowledge of the theory of the architecture. Students see how the different parts of the concepts are all working together to solve a bigger problem. I implement this part by using the Role Playing Pattern and/or the Physical Analogy Pattern (see Related Patterns for more information).

Lab: Students complete an exercise prepared by the instructor in order to figure out how the concept works. The instructions for the lab should be very detailed first with less detailed

explanations while it's going on. The first step in the lab is more like an exercise in which the students fill in the gaps. In the next step it is more like a laboratory because the students explore the system themselves.

Discussion: This should be a student-centric discussion where the students recover what they have seen during the lab phase. You should lead the discussion towards the next problem (often the students come up with the problem themselves). In the discussion phase it's your responsibility to "make" this pattern really incremental.

SAMPLE SETTING

Problem statement: Show a problem out of the students world. If you don't have any in mind just take one in which the problem idea is easily grasped. Imagine you want to develop a UI application, where you have to define some business objects and you want to view these business objects in different UIs for different users, e.g. take a bar chart, a pie and a table. You can interact with every kind of view, e.g. fill in the table, enlarge a bar with the mouse, etc. Every change in the view should change the model and every change of the model should be seen in every view, so we need something like a dependency mechanism.

Lecture: Explain which classes build the MVC framework and how the dependency mechanism works.

Role Play: Here we need at least one student acting as the model, one playing the view and one for the controller. Messages go back and forth by throwing a soft ball between the students.

Lab: Present the students first with a completed source example and let them examine the code with the debugger. Now they can see how the messages are sent and implemented in the "real world". A more difficult example would be to let the students develop a small application for themselves.

Discussion: Staying with the MVC example, the students would probably state that MVC seems to be a rather complicated thing for developing just a small application, or believe that a lot of code has to be reproduced for every application. See, we have the need for the concept of ValueModels, which will start the next iteration of performing the pattern.

DISCUSSION

This pattern shows students why somebody has introduced a particular software concept. Understanding the *purpose* of the concept motivates the students to know more about how this specific problem is solved. This leads to a positive learning environment, which is the best learning motivation you can get: enthusiasm.

The discussion phase should lead *incrementally* into the next problem statement, so it's easy to introduce the next, more complex concept. For the students it's easier to grasp the complex structure, not only because the simpler framework is often repeated by the more complex one. This way the knowledge is strengthened by repetition.

The students will obtain a better understanding of complex software concepts by giving them the chance to behave as the objects being involved in a role play. Teaching is certainly only successful if the new way of thinking is trained and exercised by *active learning*.

EXPERIENCES

You have to control the discussion so it leads to the next problem statement. For instance, go from MVC - ValueModel - AspectAdaptor to AspectAdaptor with subjectChannels.

I made good experiences with clarifying the problem domain, so that every student can identify the problem from his/her own experience. Do not try to discuss different solutions, since the goal is to introduce a specific concept like MVC and not what else could have been developed. On the other hand, the students will get frustrated very soon if they see that their solution is not of much interest.

The lecture or theory itself should be as short as possible so you can start the role play as soon as possible.

Ensure that the students change the roles in every increment.

Play the game as close to reality as possible. Omit material which has already been clarified and therefore just prolongs the play. For example, explaining method lookup is not necessary. Don't be afraid of playing the game. My experience is that, especially in industry, the students are happy if they get a less abstract explanation, even in a "real serious" environment like a bank.

Make sure the lab leads step by step from more basic things in the concept to the more complex ones.

CULTURAL DEPENDENCIES

In the audience are handicapped people, who are unable to participate in an active role play as described in the sample setting.

RESOURCES NEEDED

Depending on the audience size and on your setting - several softballs and a white board.

EXAMPLE INSTANCES OF THIS PATTERN

I have used this pattern to teach various aspects of Visual Works, such as MVC, ValueModels, ApectsAdaptors, and SubjectChannels. To teach the SubjectChannels I found it useful to simplify the architecture and use one student as the widget, one as the AspectAdaptor, and one as the model. But the model sits on a subjectChannel, here on a chair. In this way I can change the model by placing a different student on the chair.

I have also trained other trainers to use the pattern and some have incorporated it into their repertoire. Joachim Schrader from BERATA GmbH for instance has used this pattern in the same context as described above.

Richard Steiger from Ensemble Soft teaches the Java Foundation Classes with this pattern. There he teaches how to build complex GUI applications with the help of a framework, called UIF - User Interaction Framework. He explains the architecture of the application by incrementally adding more and more features, picking a problem to solve at each stage.

Paul Dyson from Cumulus Systems Ltd. has used this pattern to teach the final-year Software Engineering course in the Electronic Systems Engineering degree at the University of Essex. He specifically used role play to describe the interactions of an MVC framework implemented in C++. What made it iterative was that they did one role-playing session before the students started on the project and one after it where they would demonstrate the interactions of their specific instantiation of the framework.

RELATED PATTERNS

Physical Analogy Pattern from Phil Mc Laughlin is the basis of *Incremental Role Play*. The focus of this pattern is: Students new to Object Technology are frequently told about modeling objects in the 'real world'. Physical analogy helps to demonstrate key concepts and presents another metaphor to students as a mental cross reference for the topic. The students themselves form the analogies to the objects, so the experience is even better memorable. On the other hand, unlike *Incremental Role Play*, *Physical Analogy* is not restricted to a certain audience size..

Concrete to Abstraction Pattern, submitted by Ian Chai, may help you to find a path for your increments. This pattern has the focus on keeping students' interest when introducing them to their first patterns (or other abstractions.) It starts with an example to which the students can relate. After the students were taken through the example, you can point out the aspects which can be applied in other instances, like general principles in the pattern. And then finally end up with describing the general pattern itself.

You may want to combine this pattern with **Three Bears Pattern**, from Kent Beck, to help the students to accept failure as a natural part of the design process. This pattern removes the 'fear of failure' as a barrier to learning by making failure part of the goal.

The **Role Playing Pattern**, submitted by David Bellin, introduces a key technique used in CRC Card Analysis, after a team brainstorms and creates a list of candidate classes. This pattern is used in the *Incremental Role Play* in a different context.

REFERENCES

The related patterns can be found at: http://www-lifia.info.unlp.edu.ar/ppp/

Acknowledgement

I would like to thank the following people for their important input for the paper: Jens Coldewey (my shepherd), Mary Lynn Manns and Richard Steiger. Additionally I want to thank all the participants of the Writers Workshop at EuroPlop 1998 in Kloster Irsee, Germany.

ANNOTATION

This pattern belongs to the category of pedagogical patterns. These reusable pedagogical design patterns provide an ability to communicate proven solutions to common problems in teaching.