Incremental Role Play

Jutta Eckstein[1]
Integral Development GmbH, 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.

*[JC1]*Chinese Proverb

THUMBNAIL

The complexity of object designs is[JC2] 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 concepts?

INDICATIONS

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

CONTRAINDICATIONS

You don't feel comfortable in performing role plays, 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. It is still doable in a larger environment, but the audience will have to be divided into smaller groups.

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 they never had.

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

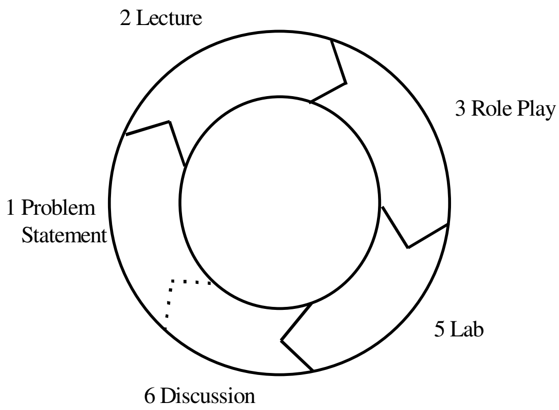
You'd like to provide a positive learning environment, but the difficulties in understanding complex concepts may frustrate the students.

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

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

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 the time. The pattern starts introducing a small framework, explains a medium framework in the next increment and concludes with teaching a large framework. 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 frameworks.

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: fascination.

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*.

STRUCTURE

- **Problem statement:** Make the students attentive to the problem this architecture is trying to solve.
- **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 on the machine 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. You should lead the discussion towards the next problem (often the students come up with the problem for 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 out the table, grow 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.

EXPERIENCES

- 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.
- Play the game as close to reality as possible. Omit material which is already 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.
- 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.

CULTURAL DEPENDENCIES

None identified yet

RESOURCES NEEDED

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

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.
  - Mr. Schrader from BERATA GmbH has used this pattern also in the context of VisualWorks
  - Richard Steiger from Ensemble Soft teaches the Java Foundation Classes with this pattern
  - Paul Dyson has used this pattern

RELATED PATTERNS

- **Physical Analogy Pattern**[2] is the basis of *Incremental Role Play*. The students themselves form the analogies to the objects, so the experience is even better memorable. On the other hand, in opposite to *Incremental Role Play*, *Physical Analogy* is not restricted to a certain audience size.[JC3].
- **Concrete to Abstraction Pattern** may help you to find a path for your increments.[3]
- You may want to combine this pattern with **Three Bears Pattern**[4] to help the students to accept failure as a natural part of the design process
- **Role Playing Pattern**[5] is used in a different context.

REFERENCES

Related patterns could be found at: http://www.cs.unca.edu/~manns/oopsla.html

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

---

[2] submitted by Phil Mc Laughlin. The focus of *Physical Analogy* 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.[0]

[3] submitted by Ian Chai, the *Concrete to Abstraction* 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[0].

[4] submitted by Kent Beck. The *Three Bears* pattern removes the [0]"fear of failure" as a barrier to learning by making failure part of the goal.

[5] submitted by David Bellin [0]. The Role Playing pattern introduces a key technique used in CRC Card Analysis, after a team brainstorms and creates a list of candidate classes.