VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui

School of Mathematical and Computing Sciences
Computer Science

Techniques for Active Learning of OO Development

Robert Biddle, James Noble, Ewan Tempero

Technical Report CS-TR-01/3
March 2001

School of Mathematical and Computing Sciences
Victoria University
PO Box 600, Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Email: Tech.Reports@mcs.vuw.ac.nz
http://www.mcs.vuw.ac.nz/research

VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui

School of Mathematical and Computing Sciences
Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341, Fax: +64 4 463 5045
Email: Tech.Reports@mcs.vuw.ac.nz
http://www.mcs.vuw.ac.nz/research

Techniques for Active Learning of OO Development

Robert Biddle, James Noble, Ewan Tempero

Technical Report CS-TR-01/3
March 2001

Abstract

We describe our use of active learning techniques to teach OO development. We have developed new techniques, adapted from CRC cards, to teach use cases for requirements gathering. We have also adapted CRC cards to teach the principles of OO. Our approach has been tried with large university classes as well as industry groups, programmers as well as business analysts and managers.

Author Information

The authors are lecturers in Computer Science at Victoria University of Wellington, New Zealand.

Techniques for Active Learning of OO Development

Robert Biddle, James Noble, Ewan Tempero
School of Mathematical and Computing Sciences
Victoria University of Wellington
Wellington, New Zealand
(robert,kjx,ewan)@mcs.vuw.ac.nz

ABSTRACT

We describe our use of active learning techniques to teach OO development. We have developed new techniques, adapted from CRC cards, to teach use cases for requirements gathering. We have also adapted CRC cards to teach the principles of OO. Our approach has been tried with large university classes as well as industry groups, programmers as well as business analysts and managers.

1. DESCRIPTION OF THE EXPERIENCE DOMAIN

In recent courses presenting object-oriented development we have introduced more active learning. This brief paper outlines some of the techniques we included, and comments on our experience.

Active learning includes a variety of techniques, but the main idea is active involvement of learners. An overview of this area of education is provided by Bonwell and Eison [4]. Learning and design are strongly related, both because design can be difficult to learn, and because design as an activity itself has similarities with learning, requiring exploration, problem solving, and evaluation.

The courses we are discussing are introducing object orientation (OO) to students with little or no background in the area. Some courses have been at our university, and other courses have been for industry groups. In some cases, the students have been programmers, but other courses have involved managers or business analysts. We focus on analysis and design, and do not address implementation details at the same time. We have typically focused on software development, but have also applied the techniques to object-oriented development of business processes. Some of the university courses have been large, with lecture sessions of 140 students.

Our starting points were two well-known techniques for OO design, CRC cards and roleplay [1, 9, 2]. These techniques are already valued for their ability to highlight issues in OO design, and their active nature also makes them very useful in teaching and learning. CRC card design requires work in small teams, and the CRC cards are concrete artifacts that support active engagement as they are created, arranged, modified, and so on. Roleplay necessarily involves each team member participating and having to really engage with the design work. However, we wanted to introduce more active elements in other activities involved in overall system development, especially at earlier stages.

2. DESCRIPTION OF THE EXPERIENCE

2.1 OO Principles

The students in these courses are newcomers to OO, so we begin by introducing the basic principles. The core ideas are objects and classes, behaviour, encapsulation, and collaboration. We also introduce basic UML at the same time, class diagrams and sequence diagrams. We then want the students to get an idea of how an OO system actually works, and we want to check their understanding.

The active technique we introduced was a kind of scripted roleplay. We introduced a simple design using a class diagram. The design is carefully chosen to be small but include an number of critical features including aggregation, inheritance, and a collection class. (We typically used a small banking system with two kinds of account.) We then outlined a situation, and presented a sequence diagram that detailed how the system worked in this situation. We introduced the idea of roleplay, describing how each person played the role of an object. To remind them of their roles, we handed out index cards with the object and class names. The students then walk through the situation according to the sequence diagram.

This active exercise worked very well, for several reasons. The active nature required involvement for everyone, so everyone became alert. The sequence diagram provided enough detail, but also required some insight from the students. As we facilitated, there were many opportunities to remind them about the OO principles, and their implications. As well, the students focused carefully on the diagrams, and picked up details of the UML they had missed in our initial presentation.

As we have repeated this in various courses, we made some minor modifications. Initially, we had distributed blank cards, assigned the students to be certain objects, and asked them to write the object and class name on the card. However, this was cumbersome, and the students often got object and class names confused. Later, we pre-prepared the cards to make sure the names were correct. This also saved

Figure 1: Using textual analysis with a scribe at an overhead projector.

time, and allowed us to hand out roles at random. It also allowed sleight of hand, meaning that we could sometimes assign more busy roles to students who need to participate more!

There are some factors to consider in choosing a design for such an exercise. One is that the example domain should be familiar and motivating to the audience. Another is that it should have enough objects for everyone to have a role. However, we accommodate some extra people as "critics" who have the special task of checking each interaction, and ensuring correct continuity. With a large audience, we speculate many could be objects within a collection, with only a few actually participating.

One final note is that we found that the quality of the design was not absolutely critical, because even poor design elements give something to talk about, and allow the OO principles to be reviewed.

2.2 Domain Analysis

After introducing OO principles, we then discuss how OO systems are specified. We begin with domain analysis and then move on to use cases.

To add some active elements to domain analysis, we introduce simple textual analysis and brainstorming through active exercises. We typically do this in a large group, with one person acting as scribe at an overhead projector (see figure 1). For textual analysis, we start with a transparency with a text description of the domain, and students first identify nouns, which we then filter and prioritize. After introducing use cases, we then introduce similar large group active exercises to identify use cases from textual descriptions by looking for verbs.

Although these exercises are very simple, their impact can be considerable. Nouns are identified as suggesting candidate objects, and verbs are identified as candidate use cases. But as the scribe underlines these on the overhead transparency, it is often possible to see a sudden understanding on the student faces. The large group atmosphere, the focus on the projected text, and the gradual annotation all seem to create a helpful environment.

Some care is needed in crafting a textual description. On the one hand, it must lead to a reasonable identification. On the other hand, it should not be overly explicit, must allow include some diversity in vocabulary and grammar,

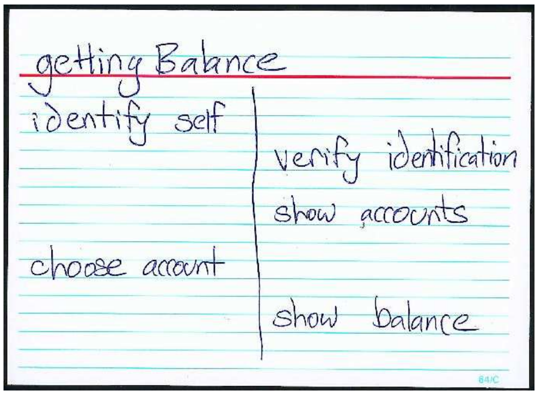

Figure 2: An example "use case card" for an essential use case, showing a dialogue between actor intentions on the left, and system responsibilities on the right.

and should even include a small amount of misdirection.

2.3 Use Cases

With candidate objects and candidate use cases in mind, we discuss actors, and introduce how to detail use cases. We introduced "essential use cases" [5]. In these, the detail of interaction is present, but the details are given in a high-level way, remaining technology-free. As well as allowing implementation using various technologies, this avoids lengthy digression, and allows faster development. (Essential use cases also have other significant advantages we discuss in a later section.) After introducing the key ideas and showing some examples, we then ask the students to work in small groups to detail some use cases.

We introduce an active technique for them to use, again based on cards and roleplay. We call the cards "use case cards", and write the name of the use case at the top, and divide the card with a line down the middle, as shown in figure 2. On the left we detail the "actor intent", and on the right we detail the "system responsibility". The result is a written dialog, where the lines on the left interleave with the lines on the right. Two people can read the parts in this dialogue, in what we call "use case roleplay" (see figure 3).

Use case cards and roleplay work for much the same reasons familiar from experience with CRC. The cards are concrete artifacts that can be manipulated, and the roleplay involves people who identify with the points of view. The roles in an essential use case stress "intent" and "responsibility", and these concepts appear to intensify the viewpoints, because the role player must examine their motivation more deeply.

With use cases, there are only two roles, actor and system. However, the very tangible dialog nature makes for a story atmosphere that facilitates critical listening, so other team members can act as critics, sometimes offering alternatives. As with CRC, the use case roleplay is also useful for presenting the design to others. After teams have developed their use cases, we typically re-assemble, and a pair for each team play out their use cases, and the larger group listens critically.

One beneficial consequence of the abstraction in essential use cases is that the behaviour of the system is considered

Figure 3: Students performing a "use case roleplay" in front of the larger group.

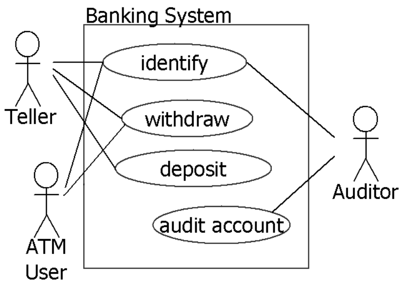

Figure 4: A use case diagram, explicitly showing the boundary around the ***system object.***

with its implementation. This makes is possible to present the system as a "black box" or *system object.* We have found this approach very useful in addressing how to determine the boundaries of the system: what it does, and what it does not do. We have found use case diagram useful in reinforcing this idea, and as shown in figure 4, we draw an explicit box denoting the boundary of the system. This convention was used by Jacobsen [7] but does not typically feature in use of UML.

The use case cards are also useful in the longer term, as we suggest keeping them to drive more detail design. The cards are easy to understand, and the left-right split works especially well in distinguishing the actor from the system. When we introduce internal design of the system using CRC, we often discuss how the later design creates the overall system behaviour as seen on the right hand sides of the use case cards.

One final observation is very significant. While we have discussed the details of our technique, it has one important effect beyond any details. The active focus on use cases stresses the importance of requirements, and implants it as a key part of the development process. Too often students first learning about system development are simply given specifications, and miss the critical importance of determining requirements.

2.4 Later Activities

In later sessions in our introductory courses, we keep the

Figure 5: A student team using the CRC technique during a breakout from a large lecture.

active involvement throughout. After our introductory sessions, and our sessions on domain analysis and uses cases, the next session directly involves CRC.

Our use of CRC is reasonably conventional, but we do play up the role of objects more than classes. In fact, although we keep and explain the term CRC, we describe the approach as concentrating more on ORH: objects, responsibilities, and helpers. We keep responsibility at the center, especially pointing out how the word responsibility implies both authority and duty, allows collaboration or delegation, and so is a useful heuristic for distribution of intelligence.

In our teaching of design, we also make use of an important characteristic of essential use cases. The system responsibilities detailed in essential use cases are the same kind of responsibilities used in CRC. Ultimately, CRC should deliver a design of collaborating objects with responsibilities that together match the responsibilities of the use cases. This means that essential use cases help in choosing object responsibilities, and also in checking responsibilities: they offer operational guidance, and traceability.

In following sessions we stress evaluation of designs, and have team exercises using heuristic evaluation following the practice of usability evaluation [8], and also look for potential change situations to consider their affect on the design (similar to the approach of Ecklund et al.[6]). We even introduce both frameworks and patterns, and integrate active exercises in our presentations. These topics conclude our introductory courses, and we believe that longer exercises and mentoring should be the next steps.

2.5 Other Aspects

We integrate the major active elements into our course in a reasonably conservative way. We still present topics in a standard lecture or seminar format. However, such presentations only last 30 minutes or so, and we then break into design teams for a quick exercise. We move about and ensure teams stay on track, but we try to keep a low profile. Between presentation sessions, we ask the students to undertake similar but longer exercises as teams.

Even in our large lecture session with 140 students, we have organised breakouts with a large number of teams, some one whom stay in the lecture theatre, while others colonize areas nearby. This has been helped by having a cafe nearby, which supplies convenient tables for students to work around, aside from supplying coffee. A further advantage of cafes in our city is that they frequently have racks of postcards suitable for use as use case or CRC cards; they are provided free because the cards feature colorful advertisements on the front, but backs are largely blank, and are even conveniently ruled down the middle.

We generally review the team design efforts as a large group, choosing which to present as time allows. The teams present their own work, and perform roleplay for the larger audience. Expecting this acts as an incentive for teams to focus on the work, and everyone looks forward to seeing what other teams have come up with. We try and allow comments on both process and results, and invite comments by team members and audience.

We find a document camera and video projector very useful for quickly displaying design documents and cards. Without these, we distribute overhead transparencies and pens, and use an overhead project for teams to present documents. Some teams initially make efforts to produce beautiful neat diagrams, but we stress that in these early stages of development it is important to focus on thinking, not on presentation: messy is good!

It is clear to us that the active learning approach has great benefits, but there is one large caveat. Active learning does require a positive attitude in the audience and significant cooperation. We have found it important to foster the right attitude continually. We explain explicitly that these techniques often involve exploration and incremental improvement, and that all kinds of ideas and suggestions can be involved in the steps along the way to a good design.

We also take a very positive attitude ourselves, and try and create an atmosphere of exploration and excitement. We liken the roleplay sessions to "actors studio", and remind people that the idea is to help us all concentrate on the design issues at hand. Early in the course, we make some early physical analogies with audio equipment [3], and use this as an excuse to play music. We then play music quietly during team exercises, and loudly to signal coming back together as a larger group. Selecting music acceptable to everyone can be a challenge, but even disagreements can help lighten the atmosphere! We have found more abstract quickly paced music to work best overall.

3. SUGGESTIONS FOR USE

In this paper we have outlined some of the techniques we developed to include more active learning techniques in our courses that introduce object-orientation. We have concentrated here on techniques early in system development. We found each of the techniques has been worthwhile, and only made small adjustments in many successive uses. Our experience has been very encouraging, and student evaluations have been very positive. We believe all these techniques should be valuable and easy to use by other educators.

The connection between learning and design is important for two different reasons. The first is that design is difficult, and learning how to design is itself an important issue. The second is that design itself is an activity with important similarities to learning. Design involves learning about the domain, and about requirements. Many important design techniques involve attempting solutions through use of heuristics, evaluation of the result, and iteration to improve. The CRC technique, for example, was proposed for learning design, but found widespread application as standard design practice. Some of our active learning techniques really did focus just on learning, such as the initial object roleplay. However, other techniques, especially the use case cards and use case roleplay, seem to have potential as reasonable techniques for system development.

4. REFERENCES

[1] Kent Beck and Ward Cunningham. A laboratory for teaching object-oriented thinking. In *Proc. of OOPSLA-89: ACM Conference on Object-Oriented Programming Systems Languages and Applications*, pages 1–6, 1989.

[2] David Bellin and Susan Suchman Simone. *The CRC Card Book*. Addison-Wesley, 1997.

[3] Robert Biddle and Ewan Tempero. Explaining issues in software reusability via an audio system analogy. In *Proceedings of the Second Australasian Computer Science Education Conference*, pages 71–77, Melbourne, Australia, July 1997.

[4] Charles C. Bonwell and James A. Eison. *Active Learning: Creating Excitement in the Classroom*. ASHE-ERIC Higher Education Reports, The George Washington University, Washington DC USA, 1991.

[5] Larry L. Constantine and Lucy A. D. Lockwood. *Software for Use: A Practical Guide to the Models and Methods of Usage Centered Design*. Addison-Wesley, 1999.

[6] Earl F. Ecklund, Lois M. L. Delcambre, and Michael J. Freiling. Change cases: use cases that identify future requirements. In *Proc. of OOPSLA-96: ACM Conference on Object-Oriented Programming Systems Languages and Applications*, 1996.

[7] Ivar Jacobson, Mahnus Christerson, Patrik Jonsson, and Gunnar Overgaard. *Object-Oriented Software Engineering*. Addison-Wesley, 1992.

[8] Jakob Nielsen. *Usability Engineering*. Academic Press, New York, 1992.

[9] Nancy Wilkinson. *Using CRC Cards - An Informal Approach to OO Development*. Cambridge University Press, 1996.