Teaching Semantic Aspects of OO Programming

Martin Blom, Eivind J. Nordby, Anna Brunstrom
e-mail: {Martin.Blom, Eivind.Nordby, Anna.Brunstrom}@kau.se
Computer Science, Karlstad University
S-651 88 Karlstad, Sweden

Abstract

One important aspect when teaching OO technology is the semantics of programming. Much time is traditionally spent on syntax and language mechanisms whereas the semantics is given less time. To remedy this problem, we try to introduce a semantic thinking throughout the entire computer science education. We have developed a contract-based programming method to enforce the semantic aspects and have performed course experiments to see what advantages such a method can have on the students' abilities. This paper presents the method and an experiment performed in a course on project work and Java to compare the method to a standard programming method. The contract-based method was positively received from the students who reported that working with the method felt natural. The results of the experiment show that the work satisfaction is slightly higher when using this contract-based method. They also show that there is a gain in the time spent on the assignment when the contract-based method is used. The long-term educational effects of the method are currently under investigation.

1 Introduction

Teaching object oriented technology incorporates teaching the semantic aspects as well as the purely syntactical aspects and language mechanisms. The semantics of operations on objects are similar to those found in imperative programming. In addition, the object concept introduces object states with semantic implications that have to be well guarded in order to produce good OO software. Every operation on an object needs to respect the rules set up for that object and not put the object in an undefined state. To guard the object from semantical misuse, so called contracts can be set up for individual operations. These contracts consist of pre- and postconditions, as derived from Hoare logic [Hoare72] and the work by Bertrand Meyer [Meyer88]. The traditional approaches for using contracts demand knowledge of formal methods, or a specific programming language or tool such as Eiffel [Meyer92] or iContract [Kramer98] and has thus never influenced the programming community to any larger extent. We believe that if using contracts was made easier and more efficient, more programmers would adopt contract-based programming as their standard way of programming. This in turn would bring a focus on semantic issues into their normal way of thinking and working. Therefore we have developed a programming method that is based on earlier formal or semi-formal work, but is aimed at the mainstream program development industry where most of our students tend to work after graduation [Blom00]. It takes into account that, for this target group, time to market is one of the most important factors along with high quality. The method can be applied in most languages with a minimum of overhead and the rigor has been relaxed such that it can be used efficiently even when the use of formal methods cannot be justified. We are working with an introduction of this method into our normal curriculum and so far it has been adopted by the first three programming courses along with a number of higher-level courses.

This paper presents the method and the course where the method was first introduced. In this course the method was compared to a standard programming method based on exceptions. We wanted to evaluate how students perceived working with semantics and also to evaluate if there were any differences in the time consumption and product quality.

2 A Method for Semantic Description

This section presents the main ideas of our proposed contract-based method. The method is based on earlier work done by Bertrand Meyer and has the following properties:

- Semantic focus - Syntactical aspects are usually well handled by editors and compiler
- Language independent - The focus is on describing rather than on executing the semantics of a program. The goal is to make the programmer think about semantics while working with program design and implementation.
- Easy to learn - The method is rather easy to learn because no prior formal knowledge except simple boolean algebra is needed. After one lecture on the method, the programmer should be able to apply most of the rules in the method guidelines without problem.
- Efficient - The method should not impose any large overhead on the development time. Some extra effort might be needed during design but this should be compensated for by less time needed for implementation, testing and maintenance.

We believe that these properties will make the introduction and adoption of the method easier and more attractive for both educational and industrial purposes.

The main focus of our method is the use of contracts, i.e. specifying pre- and postconditions for every function and respecting these. All data elements of a module, i.e. a class, should be well described. There should be no need to read the code to understand what a data element represents.

The method description is composed of an introduction and a detailed list of quality criteria, which should be satisfied in the final product, and a list of practical rules on how to satisfy the criteria. The rule describing how to design interfaces with preconditions is for instance: "Think about what parameter values can be accepted for the function and what parameter values could cause problems or errors in that function. The precondition should serve as a filter, allowing correct values to pass and filtering away unwanted values." The most current version of the method can be found in [Blom00].

3 Testing the Method in a Course

This section presents the course in which we compared the contract-based method with a reference method. This section also presents the motivation behind the course setup, the grouping of students, the reference method used in the comparison experiment as well as the rationale for the assignment.

3.1 Course Overview

The course was given at the Department of Computer Science, Karlstad University, Sweden in 1999. The course ran over ten weeks and was worth five credits. All students participating in the course did so as part of their undergraduate studies in the computer science program, most of them in their third year. The course has since 1999 been given every year and is now part of our normal curriculum.

Because the course was based on project work, few lectures were given and most of the teaching was done in weekly meetings. One introductory lecture explained the goal of the course, four lectures gave a compressed introduction to Java and OO, one lecture focused on project work and organization and one introduced the methods the students would use. The remaining part of the course offered one 20-minute meeting per group every week to discuss project work issues, programming problems and how to use the methods. To ensure that all students received the same information, a web-based helpdesk was set up in the form of an FAQ, where all questions unrelated to the methods would be posted together with our answers. This was done because we wanted students to have access to the information as quickly as possible and to minimize the information gap that might otherwise have occurred because some groups asked more questions than others.

3.2 Motivation for Project Work

Many authors have reported on the advantages of the project work form and its appropriateness for teaching software engineering [Dawson97], [Miller98], [Keen98]. The project work form is one application of the generally applicable experiential learning methodology suggested by Kolb [Kolb84], who argued that experiments are a crucial part of the learning process. As a method of learning, students working in groups, helping each other within the group and working together on problems also includes most of the important parts introduced in the action learning approach [Revans84], [Revans92]. According to Revans, action learning is a process by which groups of people work on real issues or problems, bearing true responsibility in real conditions. Including project work in education is of course an attempt to create a situation that resembles the real world as closely as possible. The reason for us wanting project work in this particular course was that the OO design and semantics will play a larger role if more people are involved and if the assignment is large enough. Small one-person assignments can usually be completed without any method support and with minimal semantic concern.

3.3 Student groups

Since we wanted to compare our contract-based method with a reference method, we wanted to split the class into two halves with an equal number of project groups in each half. A final total of 36 students was allowed to take the course, thus forming nine four-person groups instead of the optimal eight. The reason for choosing four persons per group was that we believed that a four-person group would be sufficiently large to form a project work group and manage the completion of the assignment in time. We also wanted as many groups as possible to obtain the maximum amount of independent data for statistical purposes. This in turn led us to keep the group size as small as possible. This project work model corresponded to the "Small Group Project" model described by Shaw [Shaw91]. To make the groups as similar as possible, we randomly placed one and sometimes two students from the second year in each group. We then randomly selected students from the third year to complete the groups. The members of each project group were split into two pairs where each pair was responsible for different parts. The choice of how and when to integrate the two software parts was left to the individual groups.

3.4 Reference Method

Since we wanted to compare our contract-based method with standard programming, we wanted a good reference method that should represent how programs are normally manufactured. Since most Java textbooks focus on exceptions for handling errors, we chose an exception-based method as reference. The reference method was needed in order to give all students experience in working with a good program development method. Since this reference method focused on the use of exceptions as error-handling mechanism, the students were encouraged to use these whenever necessary. The exception-based rule as to how to design interfaces was for instance: "If something can go wrong in the call to a function or if the call can come at a bad time, the function should be able to raise an exception."

Both the criteria and the practical rules in the method documents were categorized according to their application in the development process. The categories were *General* for criteria that are generally applicable, *Interface* for criteria that are crucial when designing interfaces, *Internal* for criteria regarding implementation of interfaces and *External* for usage of modules through their interface. The complete list of criteria used in the methods is given in table 1. The main difference between the two methods was in their approach to error management. The contract-based method focused on error prevention, whereas the exception-based method focused on error handling. The description of both methods included a common section on modularity and design techniques, which was standard information but was still necessary in a method document. Both the contract-based method and the exception-based method helped the students to work with OO design, but the contract-based method had a stronger focus on semantics.

Table 1: List of criteria for the methods

		Method	
Category	**Criterion**	**Contract**	**Exception**
General	Modularity	X	X
General	Weak Coupling	X	
General	Encapsulation of Data and Functions	X	X
General	Data and Function Hiding	X	
General	Data and Function Abstraction	X	X
Interface	Function Description	X	X
Interface	Contracts	X	
Interface	Exceptions		X
Internal	Respecting the Contracts	X	
Internal	Data Description	X	X
External	Respecting the Contracts	X	
External	Using Functions properly	X	X
External	Handling Exceptions		X

3.5 Assignment Rationale

The factors taken into account when developing the project assignment were size, difficulty, student motivation and modular OO design. Identifying an assignment satisfying these demands was simple: a game. A suitable game idea was a classic one: a strategy game in which players start with one ”settler” and gradually expand their ”kingdom”, building cities, settling on new land and doing battle with other players. The main idea is simple, and the graphics need not be complicated, although certain limitations were introduced to make the completion of the game feasible within the given course time. The game was divided into two parts, one part handling user interface and artificial intelligence and the other part handling game engine and rules. For a full specification of the game see [Blom99].

4 Results

Both methods were surprisingly well received, considering the fact that they sometimes demanded more work than working without methods, especially when these were used for the first time. An interesting fact was that we received comments from both class halves that they did not think their method differed much from the way in which they otherwise programmed. This might be because at least half of both method descriptions were formalized ”common programming sense” and because only the semantic parts were new to them. The students also responded positively to the project work form and were happy not to have a written exam. They appreciated learning by actually working with the problems - learning by doing. The reactions to the assignment chosen for the project were also positive. The most common response was that it was fun to construct a game instead of a less appealing product. The students had been exposed to OO programming before, but a deeper understanding was purveyed in this course as the students had a large assignment where the OO design and semantics were stressed.

When comparing the results for the two methods, some differences could be found. The results of the experiment show that the work satisfaction is slightly higher in groups using the contract-based method. They also show that there is a gain in the time spent on the assignment when the contract-based method is used. The source code was analyzed for several quality metrics but no statistically significant difference was found regarding the product quality. One interesting aspect of the results is that the contract-based method did not impose any overall increase in time consumption, which might have been expected due to the increased focus on design. The results suggest that using contracts demand more work in the design phase, but that this is compensated for by less time spent on implementation.

5 Future Work

Our contract-based method is currently being introduced in a number of programming courses and the effects of that is being recorded through student enquiries. We want to measure if using the method will help the students understand semantic aspects better. We are currently also evaluating the industrial introduction of our method, but no results are available as of yet.

References

[Blom99] Blom, M., *Assignment Specification*, (in Swedish), http://www.cs.kau.se/-~martin/java/programspecifikation.html, Department of Computer Science, Karlstad University, Sweden, 1999

[Blom00] Blom, M., Nordby, E.J. and Brunstrom, A., *Method Description for Semla: A Software Design Method with a Focus on Semantics*, Karlstad University Studies, 2000/25, Karlstad University, Sweden, 2000.

[Dawson97] Dawson, R. and Newsham, R., *Introducing Software Engineers to the Real World*, IEEE Software Vol. 14, No. 6: Nov/Dec 1997, pp. 37-43

[Hoare72] Hoare, C.A.R. *Proof of Correctness of Data Representation* in Acta Informatica vol 1, 1972m pages 271-281.

[Keen98] Keen, C., Lockwood C. and Lamp, J., *A Client-focussed, Teams-of-Teams Approach to Software Development Projects*, Proceedings International Conference on Software Engineering: Education & Practice, Dunedin, New Zealand, 1998

[Kolb84] Kolb, D., *Experimental Learning, Experiences as the Source of Learning and Development*, Prentice-Hall, 1984

[Kramer98] Kramer, R., *iContract - the Java Design by Contract Tool* in Proceedings of the TOOLS'98 Conference, Santa-Barbara, USA, 1998

[Meyer88] Meyer, B., *Object Oriented Software Construction*, Prentice Hall, 1988

[Meyer92] Meyer, B., *Eiffel: the Language*, Prentice Hall, 1992

[Miller98] Miller, J. and Mingins, C., *Putting the Practice into Software Engineering Education*, Proceedings International Conference on Software Engineering: Education & Practice, Dunedin, New Zealand, 1998

[Revans84] Revans, R., *The Sequence of Managerial Achievements*, MCB University Press, Bradford 1984.

[Revans92] Revans, R., *The Origins and Growth of Action Learning*, Chartwell Brath Lty, Bromley 1982.

[Shaw91] Shaw, M., Tomayko, J. E., *Models for undergraduate project courses in sofware engineering*, Technical Report CMU-CS-91-174, Carneige-Mellon University, 1991