The Relevance of Design in CS1

Amparo López Gaona
Departamento de Matemáticas
Facultad de Ciencias
Universidad Nacional Autónoma de México
Distrito Federal 04510 Mexico
<alg@fciencias.unam.mx>

Abstract
Most of the papers on the experiences in teaching the first object-oriented programming course are biased towards the selection of the best programming language. Sometimes we argue the pros and cons of particular languages (C, C++, Java). My point is that teaching a programming methodology is the most important element for such a course.

Introduction

For several years I have been teaching CS1 at the Science School of the Universidad Nacional Autónoma de México. This course is one semester long and its objectives are: 1) To give a broad view of computer science, 2) To promote the use of computers as tools for a large class of real and potential applications, and 3) To teach program development using an object-oriented methodology as an aid in problem solving.

For this course, previous programming experience is not a requirement. Personally, I prefer students without such an experience. Those who have programmed before this course usually have bad programming habits.

This work is centered in my experience teaching a programming methodology to reach the third objective of the course. It is obvious that a programming language is needed to program, but it must be seen as the notation [5] or tool used to apply the general concepts of the programming paradigm been used—in this case the object-oriented paradigm. In my opinion, our students should not be the masters of some programming language at the end of the course (if being a master means to know all the tricks and shortcuts of the language being used). They must be able to tackle problems of some complexity using the object orientation applied to a particular language. We must inculcate our students the idea of programming as something more than just writing a computer program. Indeed, programming means to follow the series of steps involved in the solution of a problem with the aid of a computer.

Methodology

The biggest part of the course is used in teaching the basis of the object-oriented paradigm and its applications. I teach this part as follows: first there are lectures on the relevance of the object-oriented paradigm and the general concepts, such as abstraction, encapsulation, object, class, and message. Then I present a methodology for working. Finally, there is an introduction to the programming language (C++ in this case). For each language construction I want to teach, there is a problem that shows how the previous material is not enough. I want students to feel the need for something new, which will be provided by the programming language or the paradigm. For example, when I want to introduce inheritance, I propose a problem such as a payroll problem. We (students and teacher) try to solve it as usual. Then I show them that it is possible to "factorize" the common elements via the inheritance mechanism. From that moment I can make an in-depth exposition of the subject, how, when and why it should be used.

The methodology we use for program development is made by the steps involved in the life cycle of a system. The steps are: 1) Problem analysis and specification; 2) Solution design; 3) Solution coding/implementation; 4) Test and debugging; 5) Maintenance; 6) Documentation (as a parallel activity for the whole process).

The analysis and specification of problems may seem an obvious phase, specially in introductory courses because teachers explain to students exactly what the programs must do, be it a classroom exercise or a homework assignment. However, no matter how simple the program can be, it is necessary that the professor's description of the problem be with as few words as possible. If possible, it must be an ambiguous description; this is to motivate students to ask more questions about the problem. As a result of this guided analysis, the problem can be defined with precision, with detailed specification. I think it is good for students to begin with this kind of system (programs) because in the future they will have to deal with real systems.

I think that program design is the most important step in all the process because in this step it is necessary to use the concepts of the programming paradigm being used (for this part of the course I use [1,3,4]). In this phase, students must find which objects (classes) are necessary to solve the problem at hand and the way that those objects relate to each other to obtain a solution. As a first approximation towards the identification of the classes, we can make a list with all

Reviewed Papers

the nouns in the problem's description made in the previous step. It is necessary to trim that list because some nouns may be attributes of the classes. Another possibility to determinate the objects is to mimic the
objects of the real world.

Once the classes have been found, it is necessary to describe what the objects must do and what is needed to do it. In this way, the objects can interact with other objects from the same class or from different classes. In a similar way as the nouns can help to find classes, verbs can be associated with the methods.

When we have defined the classes needed to solve the problem, it is time to establish for each class, the objects they have to collaborate with to commit their responsibilities. Also, we should know what are the objects that can realize any of the operations needed for this object. In this step it is necessary to describe for each object, how is it going to commit its responsibilities. To do it, an algorithm must be defined for each method in the class. Also, it is necessary to establish the starting point for the actions of each object. The reason is that in real life where some objects are able to begin their work themselves; in the object-oriented programming world, it is necessary for an agent to start the action.

At this time we are almost done with the design, we only need to define for each class what are the functions and data that are going to be available for everyone. We must decide which ones are going to be public and which ones are going to be of internal use, the private part. It seems to be obvious that all the methods must be in the public section, as they are going to be required by other objects. Nevertheless, there are some methods that nobody but the objects in the class itself require; those methods must be in the private section. With a similar argumentation, it seems obvious that data must be in the private section because the state of the object should not be modified by other objects. The object itself is the only responsible for the manipulation of its state.

It is important to have some objects collaborating within them, not doing so will lead to a huge (and unique) object dealing with all the job [2]. As part of the design itself, and after it has been defined which are the classes needed to solve the problem, it is time to build the detailed sequence of simple steps that specify the messages that must be sent to the objects to solve the problem at hand. Although the algorithms must be simple, we use the structured programming technique to build them.

When the previous steps have been successfully completed, the codification is a simple step for the students because they only need to make a simple translation into the programming language that is being used. Every exercise is designed to motivate the use of some structure of the programming language, so the coding step is even easier. Even for the simplest programs, they must be well structured, make efficient and effective use of the programming language, and they must avoid programming tricks. The code must reflect the comprehension of the problem and must be legible and clear.

In all this process we remark on the importance of documentation. Students are required to write it for all the programs. They must document the code and the design, including the complete description of the problem. Sometimes this work is boring, especially for homework assignments. However, students must develop the feeling of an activity required for all their programs.

The maintenance of the systems is exercised as part of their homework. Students usually have to make some modifications to the programs developed in the classroom. The description of this methodology may look as a sequence of steps, one after the other, but in practice is an iterative and effective process.

Experience and Implications

For some terms I have taught this semester course in the science school with the objective of teaching a programming methodology, not just a programming language. With this, students are able to face the problems that involve the use of computers in their solution. For the practical part, the C++ programming language is used.

Teaching a programming language is not a difficult task. The difficult, and more important, task is teaching how to use the language within a methodology, being conscious of the underlying concepts of the programming paradigm. The language must be the medium for programming, not the objective. Following this approach, educators must guide students in their design, inviting them to cooperate in this interactive work. It is a complex task, specially at the beginning of the course, because the students do not have a clear idea of what is the expected collaboration.

The main difficulty for the teacher is the definition of the problems. They must be simple enough for the students to make a good description within a short time. At the same time, they must have some degree of complexity to make attractive the use of the object-oriented paradigm.

There is another difficulty: There is no rule to determinate exactly the objects involved in the solution of a problem. That is the origin of complexity of this step, specially in the first programming course because students have no experience and they are eager to write some lines of code. It is our duty as teachers to help students gain that experience.

Recently, I had an invitation from other university to help them with their object-oriented programming course. The students were in the middle of a Java course, so I worked with them applying this approach. After I have explained the methodology for the development of programs, we worked with no mention of a particular programming language. The students had a good level of comprehension for the paradigm, so they could apply these concepts to their favorite programming language when they wanted to solve their homework problems. I could verify that in the introductory course to object oriented programming, the most important part is to know and have a full comprehension of the basic concepts of the paradigm, not the programming language. Students must define, design, and code the classes needed to solve a given problem.

Conclusion

The most difficult part in all the programming process is how to define exactly what do you want to do. It is equally difficult to define how to get the solution to the problem at hand because there is no precise rule; all you have are heuristics. Because of this, you need to be creative, patient, and have some experience. Students acquire the latter as the course flows by applying the proposed methodology.

As a summary, I recommend to work with emphasis in the design of the programs, this is richer than the mere teaching of a particular programming language. As a bonus, once the students have learned how to solve problems it is a simple task to change the programming language. In my opinion, there must be a course in program design before teaching a programming language. This approach requires of additional work from the teachers, as they have to motivate the active participation of the students.

References

1. Decker, R. And Hirshfield, S. *The Object Concept (An introduction to computer programming using C++).* International Thompson Publishing Company, 1995.
2. Holland, S. And Woodman M. *Avoiding Object Misconceptions.* ACM SIGSE, pp. 131-134, 1997.
3. Lippman S. *C++ Premier.* Addison-Wesley, 1993.
4. Stroustrup B. *The C++ Programming Language.* Addison-Wesley, 1993.
5. Taivalsaari A. *On the Notion of Inheritance.* ACM Computing Surveys, Vol. 8, No.. 3, pp. 438-479, 1996.