ISBN 978-83-60810-51-4

Smalltalk: the Leading Language to Learn Object-Oriented Programming

José A. Gallud, Ricardo Tesoriero and Pedro González
ISE Research Group. Computing Research Institute.
University of Castilla-La Mancha, Campus universitario
s/n, 02071 Albacete, Spain Email: jose.gallud@uclm.es

Abstract—**The use of Java in the first courses of Computing, Computer Sciences and similar degrees is widely accepted. However, many programming professors realize that while is possible for students to use an Object-Oriented language, is also possible to program with them without applying an Object-Oriented mentality. This paper defends the use of Smalltalk programming language as the best option for students to learn Object-Oriented programming and acquiring an Object-Oriented mentality at the same time. This study is based on three years of experience in a course on Software Design.**

I. Introduction

THE Smalltalk language enjoys in the programming world the same consideration as Latin language receives in the speaking communicating field, that is to say, it is considered a dead programming language.

However, this paper is going to defend the fact by which Smalltalk is one of the best programming languages to introduce students in the Object-Oriented (OO) programming world.

In particular this paper will show how students are more creative, imaginative and focused on the domain when using Smalltalk than with other programming language.

This proposal has the only purpose of improving the way students learn and put into practice the OO programming concepts and techniques. Many times, professors and practitioners realize that recent graduates do not apply the OO paradigm as it should be.

Next section presents a review of different languages used in introductory courses on Object-Oriented programming. Section III describes the Smalltalk features that allow students to acquire the OO paradigm. Section IV presents a case of study

II. Learning Object-Oriented Programming

This section is devoted to review the different programming languages that have been used in introductory courses of programming.

This work was supported by the authors

In this revision we are not taking into account the programming environments but only the OO features offered by the reviewed languages.

One of the most widely used OO languages is C++. The community of programmers is huge and its use in teaching OO was also considered a reference in the past. Programmers suffered the problems of managing pointers and they should manage the space allocation, cause of many side undesired effects.

The apparition of the Java language was quickly celebrated by the community [2]. Some of the previous headache problems were solved with the introduction of the garbage collection and a clean programming scheme. C# is also a widely accepted OO programming language but does not add any essential difference to Java.

Both Java and C# are supported by powerful frameworks, big libraries and a massive community of programmers. These languages offer a lot of opportunities to our students. However, in our experience, they are not the best option to learn the OO paradigm, mainly because students tend to focus on technical aspects rather than in finding a good OO solution to a given problem.

A recently presented language called Linq [1] tries to add some powerful mechanisms to manage collections, which help programmers to integrate SQL language in own OO language.

III. The True OO Mentality

When a student solves a problem in a way the solution is not well designed, from the OO point of view, we say that this student does not have a true OO mentality. He or she does not see his or her solution as a set of communicating objects.

This section exposes some common mistakes that undergraduate students tend to perform when they use an OO language without a true OO mentality. All errors can be put in relation with a quality factor as those defined in [4].

A first common mistake is to think in a solution as a set of functions. Moreover, some programmers try to solve the given problem by finding a root function (Main), rather than drawing a class diagram and defining objects and relationship among them.

A second recurrent mistake is to mix in the same class responsibilities and methods relative to different entities or concepts. This error makes the extensibility impossible to get.

Regarding the MVC pattern, using some frameworks is practically impossible to structure the code with modularity. Code responsible to manage GUI classes is frequently mixed with business code or data access code, all together in the same class.

At last but not least important is how the use of design patterns [3], a fundamental piece of work in the design of many software solutions, is well supported and integrated in the language mechanisms.

IV. What Smalltalk Offers

This section describes those features that Smalltalk offers to students and some other OO languages do not provide with.

I. Objects and Messages

One of the most important language features is that by using Smalltalk, the student only can manage objects and messages. There is no other possibility for them to program. In this way, when students reach the minimum language skills, they find themselves thinking in objects and only in objects.

II. Generality

There are no data types in Smalltalk. So, there is no need to define generics because all is generic in Smalltalk. This aspect could introduce an interminable discussion but it allows the programmer to focus on the problem rather than in the technology.

III.Collections

Manage collections in Smalltalk is something beautiful, smart and also funny. The way Smalltalk manages collections together with the language mechanisms called blocks allow programmers to write complex iterative expressions using over 50% less code than it is required when Java or C# is required.

IV. Access to Library Sources

The access to a set of complete and powerful class libraries in Smalltalk is something it has in common with Java and C#. The differential aspect in Smalltalk is that the programmer can exam all the libraries by accessing the source codes.

V. Dynamic binding

The dynamic binding is noted here in the sense the programmer can modify the code while he or she is debugging and so errors can be easily corrected. As all in Smalltalk is dynamic there is not any type checking (related to point II).

VI. Design Patterns

Consider the above mentioned MVC pattern. When a GUI application is developed using Smalltalk, the developer has no other option than define different classes to support the View (V in MVC) and the Model (M in MVC). Other programming environments do not force developers to act in this way and the consequence is the generation of a lot of spaghetti code. These language features force the acquisition of the true OO mentality since students only can work with objects and messages, they cannot use data types and the consequence is a quick.

VII. Smalltalk In Action

This section shows the experiences and results of many years of teaching programming courses with different technologies.

The first experience was the course “Visual Programming”, which was given in the 5th semester as an optional course of the Computing degree. This course was given by the authors during 5 years.

The second experience was in the context of a mandatory course called “Software Design” that was given in the 6th semester of the same degree.

In the “Visual Programming” course, students enjoyed the way they designed Web and desktop applications with low effort using .NET platform. However the solutions were of a low quality considering OO quality factors.

On the other hand, during the “Software Design”, students were forced to use Smalltalk VisualWorks. The first weeks, students suffered a kind of shock due to the big differences in the syntax and language mechanisms in comparison with Java or C#. However, after two weeks, students were able to design OO solutions for the problems they were asked to solve. Most of them had a true OO view of the problem and were able to manage not only the correctness or robustness but also extensibility and reutilization.

V.Conclusions

This paper proposes the use of Smalltalk as the first language for students to learn the Object-Oriented paradigm.

The paper describes what the most meaningful features of Smalltalk are and why they contribute to the adequate understanding of the OO paradigm.

Some real experiences prove the validity of the proposal.

References

[1] B. Beckman, “Why LINQ Matters: Cloud Composability Guaranteed”. Communications of the ACM. April 2012. Doi:10.1145/2133806.2133820
[2] T. Budd. “Object-Oriented Programming”. Addison-Wesley.
[3] E. Gamma et al., “Design patterns: elements of reusable object-oriented software”. Addison-Wesley. 2005.
[4] B. Meyer, “Object-Oriented Software Construction” second edition. Prentice Hall PTR, 1998.