Telematics and Informatics xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Telematics and Informatics

journal homepage: www.elsevier.com/locate/tele

Design and evaluation of an alternative programming paradigms course

Francisco Ortin *, Jose Manuel Redondo, Jose Quiroga

University of Oviedo, Computer Science Department, Calvo Sotelo s/n, 33007, Oviedo, Spain

ARTICLE INFO

Article history:
Received 29 March 2016
Received in revised form 7 September 2016
Accepted 21 September 2016
Available online xxxx

Keywords:
Programming paradigms
Object-orientation
Functional programming
Concurrency
Parallelism
Meta-programming
Dynamic typing
C#

ABSTRACT

The knowledge of the most common programming paradigms, and the basic abstractions provided by each paradigm, are competencies to be attained by Software Engineering undergraduate students. These abstractions also include the basis of concurrent and parallel programming, present in different programming paradigms. In an existing Software Engineering degree, these competencies were assigned to the Programming Technology and Paradigms course. We present the approach followed in the design of that course to teach object-oriented, functional, concurrent and parallel programming to second year undergraduate students with basic knowledge of Java. The time limitations of the course prevented us from using various programming languages. After analyzing different alternatives, we chose C# to teach the course. We describe the most important challenges faced and how we addressed them. The course success rate is slightly greater than the rest of courses in the same year and degree, while performance rates and average marks are analogous. There is no influence of age and gender on the final mark, but students retaking the course have significantly worse evaluation than those enrolled for the first time. The students' self-evaluation revealed that the proposed course has a strong influence on the achievement of the expected learning outcomes, and their satisfaction with the course was significantly higher than with the rest of courses in the same degree.

1. Introduction

A defining factor for many programming courses is the choice of a programming paradigm. Indeed, half of the six introductory course models identified in the ACM/IEEE computer science curricula explicitly refer to a specific paradigm: imperative-first, objects-first and functional-first (ACM/IEEE-CS, 2013). The first approach uses the traditional imperative paradigm, the second one emphasizes the early use of objects and object-oriented design, and the third one introduces algorithmic concepts with the functional paradigm.

Beyond their programming skills, computer science students must be able to know the existing paradigms and select the most appropriate one (ACM/IEEE-CS, 2001). The ACM/IEEE curricula also identifies the necessity of teaching concurrent and parallel programming (ACM/IEEE-CS, 2013). Therefore, the abstractions provided by the main programming paradigms and the foundations of concurrent and parallel programming are commonly included in Software Engineering degrees.

* Corresponding author.
E-mail addresses: ortin@uniovi.es (F. Ortin), redondojose@uniovi.es (J.M. Redondo), quirogajose@uniovi.es (J. Quiroga).
URL: http://www.reflection.uniovi.es/ortin (F. Ortin).

http://dx.doi.org/10.1016/j.tele.2016.09.014
0736-5853/

In the last years, dynamic languages such as Python, Ruby and JavaScript have become increasingly popular (Paulson, 2007). These languages provide high-level features at runtime that other languages only provide statically by modifying the source code before compilation (Callau et al., 2013). They support runtime meta-programming, allowing programs to dynamically write and manipulate other programs (and even themselves) as data. These meta-programming features include different levels of reflection and runtime code evaluation (Ortin et al., 2009). Due to the current widespread use of these languages, their distinguishing features of dynamic languages could also be included in a programming paradigm course.

When teaching different paradigms in the same course, an important decision to be made is the selection of one or more programming languages. If different languages are used, the students should get used to different language syntaxes in the same course (Cain, 2016). This option enriches the knowledge of the student, but requires enough time to learn different languages. On the other hand, the use of a single language would shorten the time required to teach multiple languages, but one single language may not provide all the abstractions provided by different paradigms. Other features such as portability, standardization, development environment, spread and support should be considered in the selection of the programming language(s) (Ortin et al., 2016).

The main contribution of this work is the design and evaluation of a programming course to teach the functional and object-oriented paradigms, concurrent and parallel programming, and the distinguishing features of dynamic languages. One of the main challenges was the selection of a single programming language, since the utilization of various languages seemed to be infeasible in the context of the course. That language must support the object-oriented and functional paradigms, concurrent and parallel programming, and the basic meta-programming services provided by most dynamic languages. We describe the design of the programming paradigms course and an evaluation showing the students' achievement of the expected learning outcomes.

The rest of this paper is structured as follows. Section 2 discusses the related work and the proposed course is detailed in Section 3. Section 4 describes the methodology used in the evaluation discussed in Section 5. The conclusions are presented in Section 6.

2. Related work

Although most of the programming courses are focused on one single paradigm, a few are aimed at introducing the main programming paradigms. The Programming Paradigms course of Stanford University is one example of the latter (Cain, 2016). This course is available online and, at present, there have been more than 29,000 students enrolled. The contents include an introduction to programming paradigms, the imperative, object-oriented and functional paradigms, concurrent programming, and advanced memory management features. Functional programming is taught using Scheme, imperative programming with C, C++ is used to teach object-orientation, for concurrent programming they use C, and dynamic typing and multi-paradigm programming is taught with Python. They also use assembly code to explain memory management.

Another example course is the Paradigms of Computer Programming course at the Université Catolique de Louvain (UCL) (Roy, 2016). This course gives an introduction to all major programming concepts, techniques and paradigms. It covers functional, object-oriented, declarative dataflow and concurrent programming. After this course, the student should be able to choose the right programming paradigm and write a program in that paradigm to solve a problem. They use the Mozart system that implements the Oz programming language (Consortium, 2016). Java is also used to teach the object-oriented paradigm. The course is targeted toward people with a basic knowledge of programming.

The online C# Programming Paradigms course taught by Allen (2016) uses C# to introduce a variety of programming styles. It includes object-oriented, imperative and functional programming. In functional programming, C# is used to teach higher-order functions, lazy evaluation, partial application and currying. It also introduces the foundations of asynchronous and parallel programming. Differences between statically and dynamically typed programming are also covered. The course slightly covers the limited structural and behavioral reflection levels (provided by `ExpandoObject` and `DynamicObject`). Language Integrated Query is another topic of the course.

The Programming Paradigms course of the Chalmers University of Technology (Sweden) describes the main programming paradigms (Bernardy, 2016). The course covers imperative, object-oriented, functional, concurrent, and logic programming. Different programming languages are used. C is used for imperative programming, C++ for object-orientation, Haskell is used for functional programming, Erlang for concurrency, and Prolog is the language used to teach logic programming. The student should be able to select the correct paradigm to solve one programming problem. The course also covers how abstractions in one paradigm can be translated into different abstractions in other paradigms.

3. Course design

3.1. Context

The proposed course, named Programming Technology and Paradigms, has been included in the Software Engineering undergraduate degree at the University of Oviedo. This mandatory course is taught in the second semester of the second year. The course has weekly sessions of theory (2 h per week) and laboratory classes (2 h each). Theory and laboratory classes

Please cite this article in press as: Ortin, F., et al. Design and evaluation of an alternative programming paradigms course. Telemat. Informat. (2016), http://dx.doi.org/10.1016/j.tele.2016.09.014