JOURNAL OF OBJECT TECHNOLOGY
Online at www.jot.fm. Published by ETH Zurich, Chair of Software Engineering. ©JOT, 2002

Vol. 1, No. 3
Special issue: TOOLS USA 2002 proceedings

An encapsulated Eiffel education environment, based on Web Services

Manu De Backer[1], **Guido Dedene**[1,2,3], **Monique Snoeck**[1]
[1]K.U.Leuven, Dept. of Applied Economic Sciences,
Naamsestraat 69, B-3000 Leuven, Belgium
[2]University of Amsterdam,
Faculty of Econometrics and Economics
[3]Vlerick Leuven Gent Management School

This paper explores some requirements for training/education environments for elementary object-oriented programming and discusses the applicability of Web Services for this purpose. It is shown how the encapsulated Web-interfaced nature of Web Services offers a very interesting object-oriented technology platform. This is illustrated by means of an implementation of an education environment for teaching Eiffel to first year Applied Economics Students. The benefits, but also the changes in the teaching process are discussed. The implementation is based on a unique combination of features of ISE Eiffel and Microsoft.NET.

1 INTRODUCTION

The Faculty of Economics and Applied Economics at the Catholic University of Leuven (K.U.Leuven) has a long tradition of offering information systems studies during the first year of study to all its students. For many years, Pascal was used as a programming language for this purpose. After a number of years of prototyping at a remote campus of K.U.Leuven, the decision was made to implement a new course for all Faculty students. This course is based on the Object Oriented Paradigm, and the Eiffel language. The choice of Eiffel is beyond the scope of this paper, and early experiences have been reported elsewhere [2].

The ambition level of the course is fairly basic: students should gain some understanding of the working and the construction of software. Most of the students will perhaps never program in their later career. Nevertheless, it is important that they gain an understanding of basic software constructs, also to come to a better interaction with Information Systems in a later Business Professional function.

This paper will focus on the requirements for an education environment to sup-

Cite this article as follows: Manu De Backer, Guido Dedene, Monique Snoeck: *An encapsulated Eiffel education environment, based on Web Services*, in Journal of Object Technology, vol. 1, no. 3, Special issue: TOOLS USA 2002 proceedings, pages 97–106, http://www.jot.fm/issues/issue_2002_08/article6

port the teaching of these subjects to a large scale group of students. After discussing the requirements, it is shown how Web Services can address many of the requirements put forward. The last part of the paper presents a solution based on ISE Eiffel in the Microsoft.NET environment: the Eiffel Education Environment (E^3).

2 REQUIREMENTS FOR AN EDUCATION ENVIRONMENT

Whenever universities or schools decide to add programming languages to their course, they face a number of practical problems. In a typical non-computer science Faculty, students have little or no programming experience, and the computing background is limited to surfing the Internet and editing Word documents, in general.

It is the challenge for the teaching staff to present a clear basis on the theoretical foundations of the course material, in such a way that the students can start practising afterwards. In particular, programming exercises enable learning by doing, in most cases facilitated by presenting a software education environment to the students. In practice, this often forces installations in large PC-classrooms and the production of a large amount of CDs to allow the students to practice the material on their own computers.

Installing and running - even a simple - software environment is not an easy process for the group of students indicated before. This has also a nasty side effect: teaching assistants, who are supposed to spend most of their time in helping the students with problems on the course contents and education, devote in reality a considerable amount of time on helping the students with installation problems. This leads already to the following set of requirements:

Requirement 1: *An education environment should have a very simple installation and configuration procedure.*

Requirement 2: *An education environment should be easy to use for students, based on intuitive User Interfaces.*

The teaching "goals" should be as explicit as possible. In [3] four types of goals are defined, with the implementation of the goals for this case:

- Teach a language: a subset of the Eiffel syntax [4]
- Teach a tool: the ISE Eiffelbench/EiffelStudio realizes this goal.
- Teach principles: the underlying principles of the Object-Oriented paradigm [5]
- Teach practices: by solving small realistic business problems.

After this course at K.U.Leuven, the students should have learned the basic mechanisms to understand and construct computer applications. The ambition level is

a first contact with software construction, illustrated with simple systems examples [2].

In this course, the principles of the Object Oriented paradigm in conjunction with basic programming structures are the more important elements. Nevertheless, the examples and tools should make the students aware of some of the complexity of programming. This leads to the following additional requirements:

Requirement 3: *An education environment should provide simple exercises, which stress the importance of programming structures.*

Requirement 4: *An education environment should allow the student to construct solutions, partially also on the experimental basis, including "trial and error". If the student has no idea about the solution, a capability to provide hints should be present in the education environment.*

All universities experience on-going economic pressure, which forces to optimize their staffing resources. Of course, this should be done without losing quality. This imposes the following requirements:

Requirement 5: *An education environment should support maximal automation of the process of creating, changing and distributing the exercises.*

Requirement 6: *An education environment should require minimal maintenance support. In particular it should be easy to adapt to new versions of the language tools (in this case, ISE Eiffel). Preferably the yearly maintenance of the education environment should not exceed one person-year.*

Furthermore, the K.U.Leuven has chosen for an aggressively use Internet/Intranet technology for education support. In this case Blackboard is used as a learning Portal, whereas Question Mark is added as a test platform.

Requirement 7: *An education environment should integrate in a frictionless fashion with other Internet/Intranet based education tools.*

3 TOWARDS AN EDUCATION ENVIRONMENT SOLUTION

Solution options

Starting from the choice for Eiffel, the straightforward solution might be the distribution of a standard ISE EiffelBench/EiffelStudio environment. This solution faces a number of practical problems:

- Not all students have the necessary PC-skills (e.g. to manipulate CDs).
- Many options in the tool are overwhelming for a first-year student, although they are necessary for professional use of the tool.
- Setting up Assembly of Classes in Eiffel (ACE-files) and choosing libraries to

use for compilation is not a simple exercise.

In general, the overall student skills on personal computers should still not be exaggerated for non-computer science studies.

Another option could be the development of a customized Eiffel CD, with a simple restricted graphical user interface and a set of exercises. This is a fairly static solution, leaving no possibility to add exercises dynamically. Moreover there is no integration with other Web-based education tools (such as Blackboard).

It is clear that a Web-based solution is preferable, by far. If a situation can be created whereby the student only needs a Web-browser environment, without additional installation, many of the above requirements are addressed.

Teaching specifications

The user interface of the education environment depends heavily on the type of exercises that are offered to the students. Three types of exercises were considered.

Option 1
The teacher gives a description of the exercise and leaves all the work to the students.

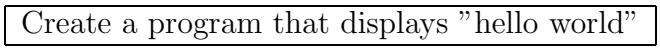

Figure 1: Exercise Assignment

Option 2
The students are given a part of a program, where they have to add some instructions. In this type of exercises it is totally up to the student to know where and what needs to be added. This may be difficult for non-computer science students in particular.

```
class
    ROOT_CLASS
creation
    make
feature  -- Initialization
    make is
         -- Creation Procedure
         do
         ...
         end
end  -- class ROOT_CLASS
```

Figure 2: Representation of the Hello World example

Option 3

The third option shows the student's programs, with clearly indicated lines where instructions must be added, of course in a consistent fashion as the following figure illustrates .

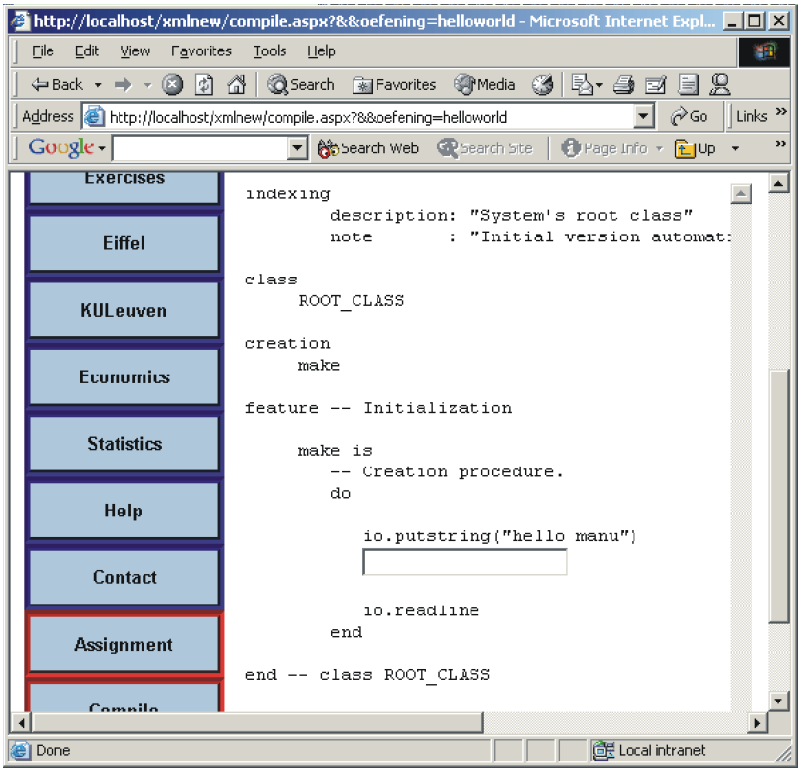

Figure 3: Representation of the Hello World example in a web site.

An education environment should largely simplify the procedure for making, compiling and running exercises. The following procedure for opening, making, changing, compiling and running exercises seems preferable:

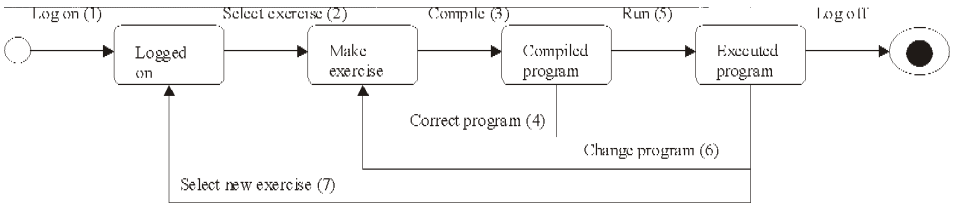

Figure 4: Preferable procedure.

4 IMPLEMENTING A WEB SERVICE BASED EIFFEL EDUCATION ENVIRONMENT

Web Services provide an ideal contemporary solution for hyperlinking software components over the Internet. The Microsoft.NET framework offers one solution for Windows-based platforms. Of course, the fact that the ISE Eiffel Compiler is largely integrated with .NET offers unique opportunities for an Eiffel Education Environment.

In the .NET framework, Web Services are enabled by developing ASP.NET (Active Server Pages). Unlike the traditional ASP, ASP.NET can rely on any programming language integrated with the .NET framework to develop Web Services. An immediate consequence of the following: once an education environment for Eiffel is developed, the same environment can - in principle - be cloned for every .NET language, including C♯, Visual Basic.NET and Object Oriented COBOL.

Figure 5 shows the collaboration of the four ASP.NET aspx-pages, which constitute this education environment. Of course, the education environment should

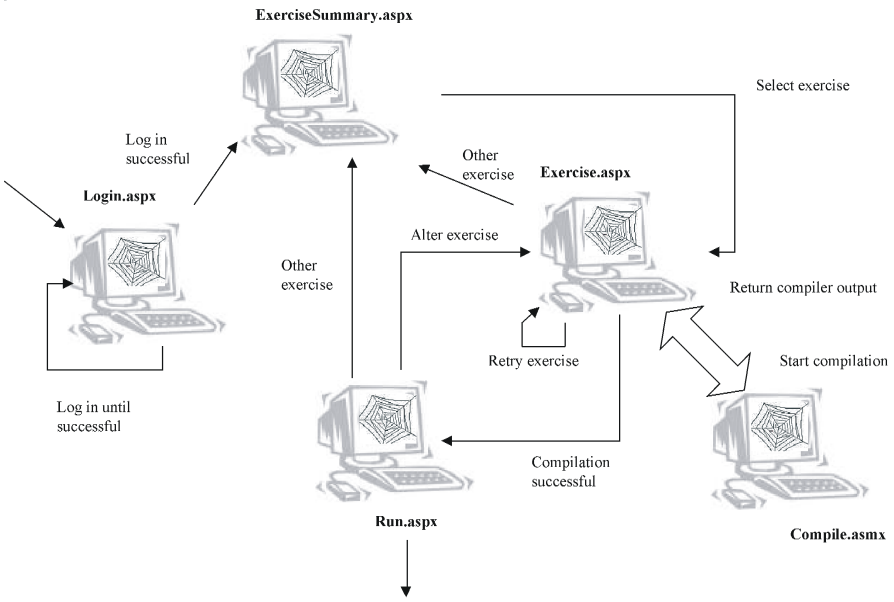

Figure 5: Preferable procedure.

provide login-facilities (**Login.aspx**). On logging in, a compilation map is created on the server for the student's exercises. Next the student is directed to the main page (**ExerciseSummary.aspx**). This page presents to the students the available exercises. The main requirement offered behind this aspx-page is the automatic ability to add/change exercises.

Every exercise consists of 3 types of files:

- One “ace-file”
- One or more class files
- One file with the description of the assignment

The main pages are build up dynamically, so a simple refresh allows to show newly added or changed exercises. By clicking on an exercise link, the student gets a Web page implementing the third interface option discussed above. It is important to realize that this is not a simple ”fill-in” form-exercise. The mutual consistency of feature names and references can only be checked by a strong typed compiler.

On this web page, the following functions have been implemented:

1. Open class files
2. Displaying classes
3. Creating blanc textboxes where needed
4. Call the compiler
5. Display the compilation error codes, when needed

The students are offered one or more classes each having zero or more blank lines. The compiler implemented is the ISE command line compiler, encapsulated in a Web Service. The Web Service is also taking care of all the environment/directory administration to isolate the different students/exercises. Figure 6 shows a screenshot of a preliminary user interface for the education environment.

The implementation of the ISE compiler error messages makes it possible to customize the error messages, to include additional references to the course material, also as present in Blackboard, and additional exercises in Question Mark.

The compilation Web Service (Compile.asmx) encapsulates the ISE Compiler environment, and realizes basically 3 functions:

- Open/Change ACE-file
- Compile
- Finish_Freezing

A unique feature of the ISE Compiler is the freezing option. This allows to create completely independent, self-contained Windows-executables. In this Eiffel Education Environment, when no errors are generated in the compilation of the student’s solution, freezing is invoked automatically, allowing the student to download the generated executable (see Figure 7).

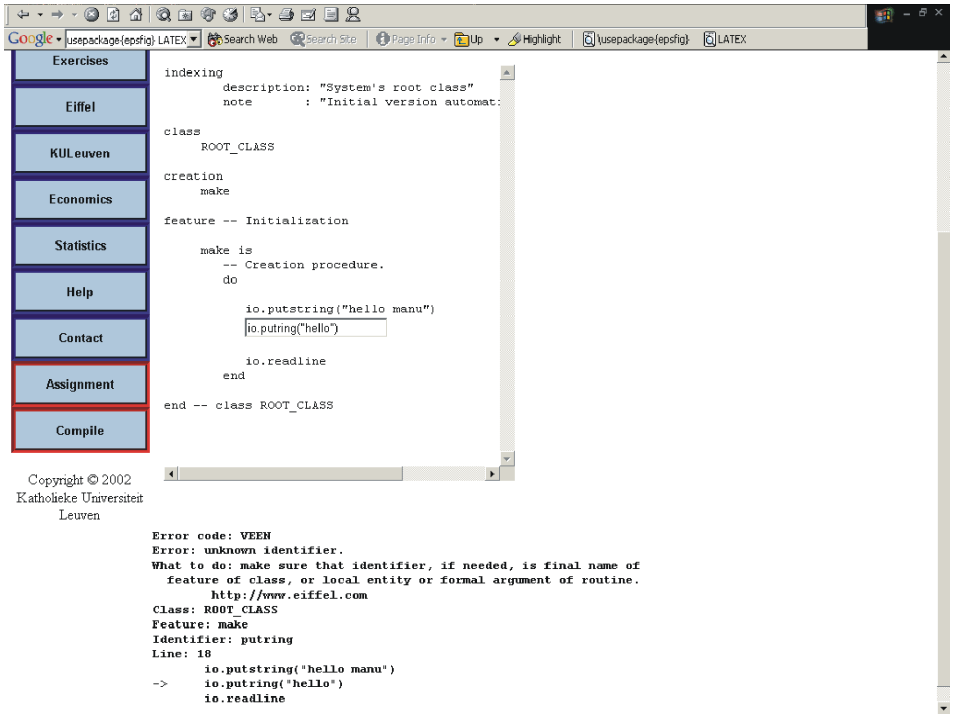

Figure 6: Compiled exercise.

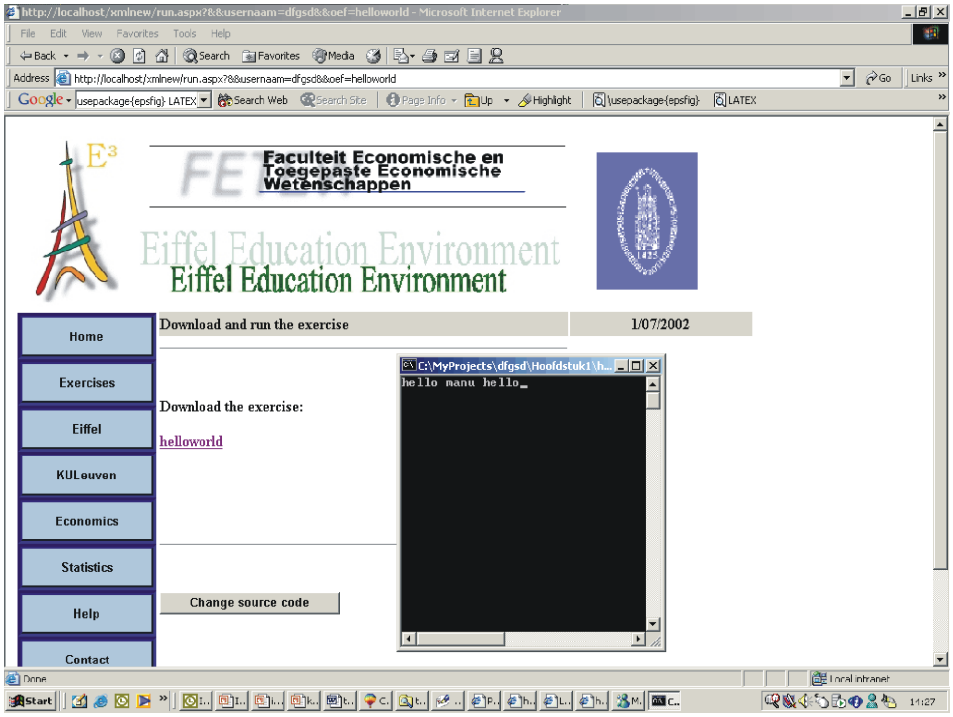

Figure 7: Run the compiled exercise.

5 CONCLUSION

This paper examined the requirements for a straightforward education environment for training non-computer science students in elementary programming. The education environment satisfies the following requirements:

- Easy installation
- Ease of use
- Simple exercises: emphasis on programming structures
- Learn by doing, trial and error, hints
- Automation of the exercise management process
- Low maintenance efforts
- Integration with other Internet/Intranet based education tools

An encapsulated education environment for Eiffel has been proposed using Web Services. The use of ASP.NET and Web Services avoids tedious installations on the side of the student. The unique compilation features of the ISE Eiffel Compiler enable better student support, and the download of successfully compiled projects.

The simplicity of the environment is striking, in particular in comparison with a typical JAVA-based education environment [1]. So far, all exercises of the introductory course texts at K.U.Leuven have been implemented and tested successfully. The environment will be volume-tested in October 2002.

REFERENCES

[1] J. Viega Alexander R. T., J. M. Bieman. Coping with java programming stress. *IEEE Computer*, 33(4):30–38, 2000.

[2] G. Dedene. Experiences teaching Eiffel as a first programming language to economy students. In *Proceedings of Technology for Object-Oriented Languages and Systems, 1999*.

[3] B. Meyer. Reflections on software education. Lecture at K.U.Leuven, 2001.

[4] B. Meyer. *Eiffel the language*. Prentice-Hall, Inc., New York, 1992.

[5] B. Meyer. *Object-oriented software construction, Second Edition*. Prentice-Hall, Inc., New York, 1997.

ABOUT THE AUTHORS

Manu De Backer is a PhD student and research assistant at the K.U.Leuven at the faculty of Economics and Applied Economics. His research interests are education environments, Eiffel, C#, .NET and intelligent agents. Email: manu.debacker@econ.kuleuven.ac.be

Prof. Dr. Guido Dedene is Full Professor Information Systems at the Catholic University of Leuven (K.U.Leuven) at the Faculty of Economics and Applied Economics. He also holds a chair on the "Development of Information and Communication Systems" at the University of Amsterdam. He is also teaching Information Systems Management at the Leuven Gent Vlerick Management School. His research interest includes formal business systems development, as well as quantitative management aspects of information systems. He publishes regularly in international journals, and obtained in 1995 the IEEE Software Best Practice Award. Prof. Dedene strongly emphasises the principles of object-oriented software engineering in his student courses. Email: guido.dedene@econ.kuleuven.ac.be

Monique Snoeck obtained her Ph.D. in May 1995 from Computer Science Department of the Katholieke Universiteit Leuven with a thesis that lays the formal foundations of MERODE. Since then she has done further research in the area of formal methods for object-oriented conceptual modelling. She now is Associate Professor with the Management Information Systems Group of the Department of Applied Economic Sciences at the Katholieke Universiteit Leuven in Belgium. In addition, she is invited lecturer at the Universit Catholique de Louvain-la-Neuve since 1997. She is and has been involved in several industrial conceptual modeling projects. Her research interest are object oriented modelling, software architecture and gender aspects of ICT. Email: monique.snoeck@econ.kuleuven.ac.be