L2Code: An Author Environment to Integrate Multiple Intelligences and Naive Bayesian Classifiers for MultiParadigm-Programming Training

M. L. Barron-Estrada, Ramon Zatarain-Cabada, J. Moises Osorio-Velazquez, L. Zepeda-Sanchez, and Carlos A. Reyes-Garcia *

Instituto Tecnologico de Culiacan, Juan de Dios Batiz s/n Col. Guadalupe, C.P. 88220
Culiacán, México, tel. +52 667 7131796
rzatarain@itculiacan.edu.mx
* Instituto Nacional de Astrofisica, Óptica y Electrónica (INAOE)
Luis Enrique Erro No. 1, Sta. Ma. Tonanzintla, Puebla, 72840, México
kargaxxi@inaoep.mx

Abstract. L2Code is an Intelligent Tutoring System used for teaching programming courses for different paradigms. It is designed to work with diverse types of modules oriented to certain ways of learning using principles of Multiple Intelligences. The learning modules can be elaborated easily by any person who teaches some topic of programming. The predictive engine uses a Naive Bayes classifier which operates in real time with the knowledge of the historical performance of the student. Results of studies show that learning characteristics can be modeled using a scheme of learning with the correct election of the attributes, establishing for that reason the primary target of L2Code.

1 Introduction

Learning a programming language is in general considered a hard task, and programming courses often have high abandon rates. It has even been recognized, that it takes about 10 years for a beginner to become an expert programmer [1]. Educational research has been made to distinguish the characteristics of beginner programmers and to study the learning process and its associations to the different aspects of programming [2, 3]. Lately also differences between procedural and object-oriented education approaches have been studied, as Java and C++ have become common educational languages [4]. Milne and Rowe [5] studied in a recent survey the difficulties of Object oriented programming by performing a web-based survey for both students and teachers.

In addition, today many students study programming languages by using distance education where they can access many tutorials in the subject. The problem of this technique of learning is the lack of an actual tutor who, according to the characteristics of the student, illustrates the learning material in a comprehensible style. In order to solve the problem, much research has been conducted by proposing

Advances in Artificial Intelligence and Applications
Research in Computer Science 32, 2007, pp. 406–416

Received 19/06/07
Accepted 31/08/07
Final version 30/09/07

using Intelligent Tutoring System (ITS) for teaching programming [6, 7, 8, 9]. However all of these works are focused on introductory programming material or are oriented to one single paradigm or programming language.

Our proposal is an ITS designed to accept diverse types of programming language paradigms oriented to different ways of learning by using the principles of Multiple Intelligences [10]. This system, named L2Code, can dynamically identify the learning characteristics of the student [11] and provide him the learning or study material according to his type of intelligence. The different programming modules can be conveniently produced by any instructor who wants to teach some programming topic. It is only necessary to specify which resources refer to which types of student intelligences, and which evaluation will be part of the different modules of the ITS. This is necessary in order to measure the student performance and to improve the prediction of the best learning resource. A predictive engine for L2Code works with a Naive Bayes classifier [12] which operates in real time with the knowledge of the historical performance of the student.

The arrangement of the paper is as follows: In Section 2, we present the architecture of L2Code describing each one of the module components. In Section 3, we discuss the implementation of several important algorithms used in the software. Some experimental results are shown in Section 4. Comparison to related work is given in section 5 and conclusions are shown in Section 6.

2 Architecture of L2Code

The general architecture of the system (figure 1) includes a set of components that allow modularization, scalability, and maintainability of the system.

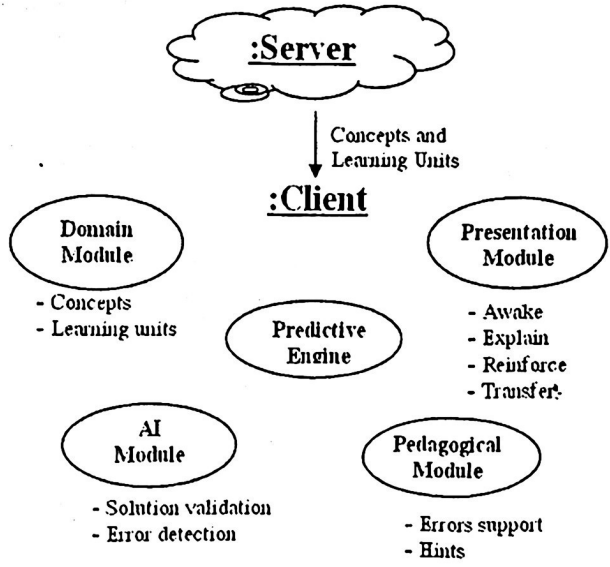

Figure 1. General architecture of L2Code

The server is the one in charge to provide the complete course that comes to be a package of different resources with its respective evaluations. The server is not more than an abstract entity, since can be distributed in internet by a Web site, or directly by the creator of the course.

The client contains the ITS. It has the following components:

- Domain Module. It is the one in charge to encapsulate the content of the course, such as concepts and learning units with their respective resources.
- Presentation Module. It is the one that works with certain unit of learning, like waking up the student, explaining some concepts, reinforcing the content or simply transferring new knowledge.
- Pedagogical Module. It is the one in charge of the tutor, making functions such as detecting errors in the answers of the student, and feed backing and guiding the student towards the correct solution.
- AI Module. Fundamental part in the operation of the pedagogical module, since it is the one that really detects the type of solution for the student, correct or incorrect, therefore the pedagogical module only worries about the feedback process.
- Predictive Engine. Its function is the one to calculate the probability that the student has taken the correct course, according to its type of intelligence measured in the degree of assimilation of the learning unit. With this calculation, the predictive engine is able to predict which would have to be the following resource that the student would have to take.

2.1 Layer Model Architecture

The layer model architecture of the system (figure 2) is defined with respect to the system scalability and maintainability. This architecture define a series of interfaces and services that are provided between each one of the layers.

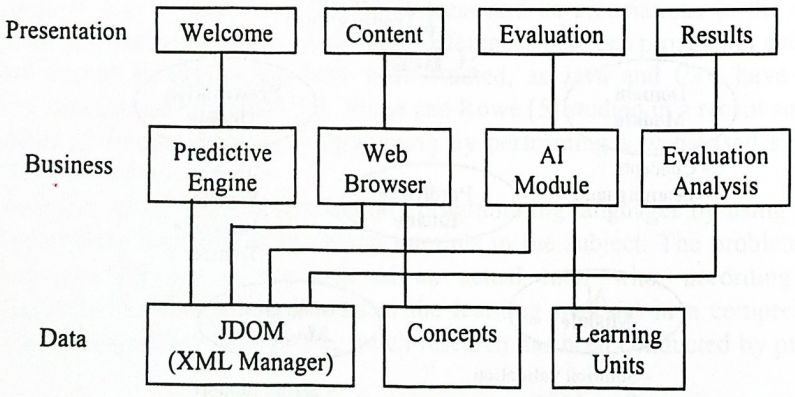

Figure 2. Layer model architecture

2.2 Learning Process in L2Code

The learning process in a module (figure 3) starts by describing basic information like name, objectives, and previous and further knowledge of the module. Next, a visualization of theoretical content is shown, and then a corresponding evaluation is performed. In this process, there exist an assistant to the student on the solution of the problems. And finally the results of the student are shown with a corresponding feedback.

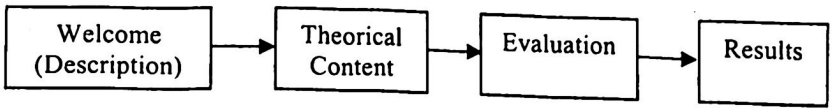

Figure 3. Learning process

2.3 Predictive Engine

As we defined previously, the predictive engine is the one in charge to calculate the probability that a student corresponds to certain type of learning resource, predicting the ideal one that the student would have to attend.

The input of the engine is composed by the results of the evaluation done to the student after the conclusion of a learning resource, and the attributes used for the evaluation, obtaining as an output the learning type of the student. This way we can indicate the appropriated resource for the student.

The following attributes have been chosen to reflect how the students use the different resources:

- Time (F, N, L). There is a range of time specified by the course creator. If the student spent less time reading the question than the low limit then *Time* is F (fast); if the time past the high limit then *Time* is L (long); otherwise *Time* is N (normal).
- First choice (Yes, No). *Yes* if the student answer is the first one he/she chose, *No* otherwise.
- Question attempted (Yes, No). *Yes* if the student attempts to answer a question, otherwise *No*.
- Accuracy (0 .. 1). Measures the approximation of the student answer with respect to the correct answer. This computation depends of the evaluation type defined by the course creator.

After determining the probability of each question, the probability corresponding to the module (resource type) is calculated considering the following attributes:

- Time (F, N, L). There is a range of time specified by the course creator. If the student spent less time reading the learning contents than the low limit then *Time* is F (fast); if the time past the high limit then *Time* is L (long); otherwise *Time* is N (normal).
- Repeat (Yes, No). *Yes* if the student had already seen this resource, *no* otherwise.
- Code value (0 .. 1). This value is defined by the course creator and says how much percentage must be assigned to code questions.

- Intelligence (VL, LM, VS, MR). It defines the type of student intelligence. According to Gardner theory [10] there are seven intelligences. We manage four of them: Verbal/Linguistic, Logical/Mathematical, Visual/Spatial and Musical/Rhythmic.

3 Implementation

The development of the system was made by following a cascade model with a modular development under the UML language [13, 14]. The system was implemented with the Java™ language [15]. L2Code makes use of two external packages that are: JDOM [16] for the XML reading and writing and SWT (Standard Widget Toolkit) [17] for the creation of natives graphical interfaces allowing all of this the multiplatform facility provided by Java™.

3.1 Naive Bayes Classifier

This algorithm (figure 4) is in charge of the probabilistic calculations to be able to make the prediction of the ideal learning resource for the student. During the interaction of the student with the learning module the attributes of this interaction are registered and, when finishing it, the corresponding probability of the actual learning resource is updated.

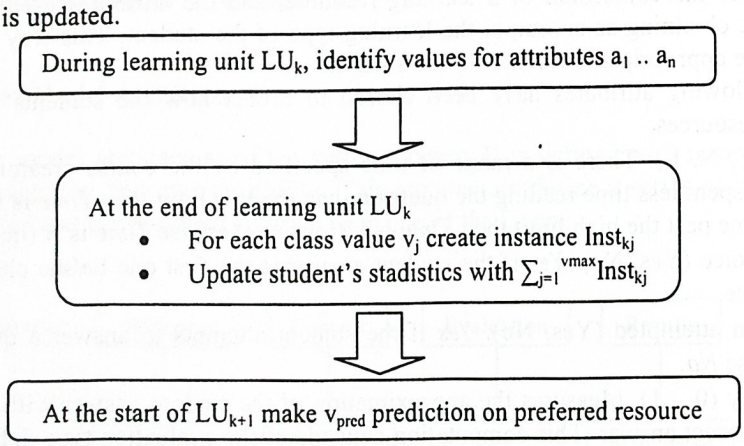

Figure 4. Naïve Bayes classifier algorithm

3.2 Evaluation Algorithms

In the process of evaluation of the learning module we define four different evaluations:

- **Multiple Options.** It offers a series of possible answers, where only one answer is correct.

- **Keywords.** Here we evaluate the answer of the student based on the amount of correct keywords that the answer contains. The algorithm is explained in figure 5.

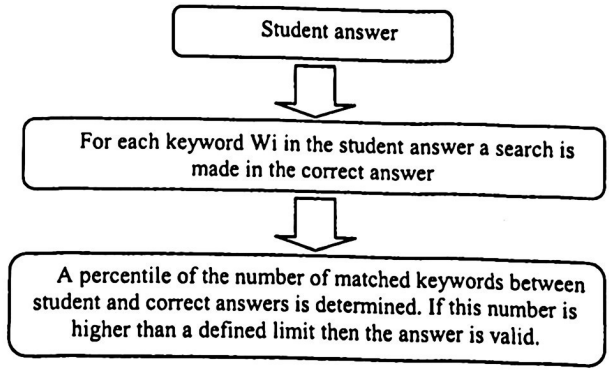

Figure 5. Evaluation with keywords

- **Edit Distance.** This is also for free answers from the student, but the evaluation method is oriented to a minimum number of characters that must be eliminated, inserted or interchanged so the answer of the student is identical to the correct answer. This is illustrated in figure 6.

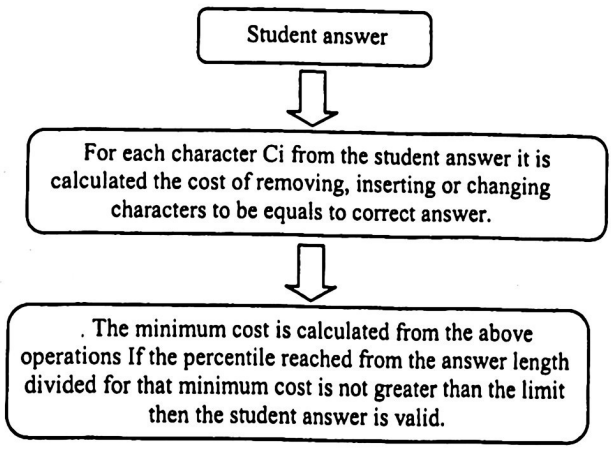

Figure 6. Evaluation with edit distance algorithm

- **Practice Evaluation (Code Problem).** This type of evaluation (see figure 7) was implemented to evaluate code and to provide hints to the student throughout its development and, at the end, a feedback of its answer.

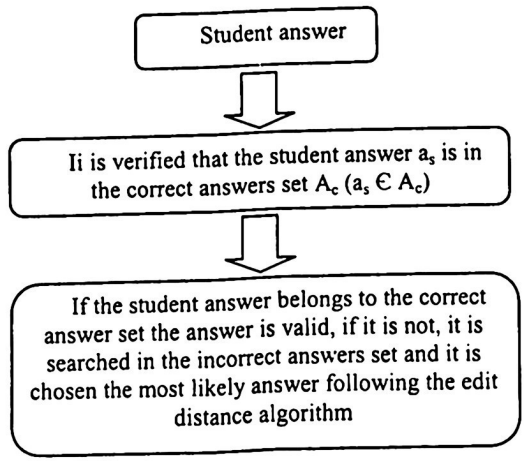

Figure 7. Algorithm for practice evaluation (code problem)

4 Experimental Results

When the student makes the election of his/her first learning module the result of the predictive engine is undecidable, since there has been no interaction with the system.

We will present an example of a student interaction with the system in a course of object-oriented programming in Java, where the student is taking the module of classes and objects.

In the Welcome section of L2Code, initial information of the learning module like objectives, description of the learning material, etc. is shown, as well as the types of presentation available.

After the student has finished the content of the module, he/she needs to be evaluated in some of the possible ways that we explained above.

When the student has finished attending one learning module and has been evaluated, a probabilistic value is determined and used for the prediction of the type of intelligence. In order to be able of comparing the final calculation with the rest of the other learning resources and to determine the appropriate resource for the student, this probabilistic value is stored and merged with the rest of the calculations made to the learning resources of the same type.

Table 1 gives an example of a student using L2Code.

Table 1. Student interaction

Student answer	Response time
methods	10
Declaration and body	35
constructor	10
True	80
"()"	20
name body arguments	80
new	25
usr = new User()	100

The interaction was in a module with Visual/Spatial intelligence type and characteristics shown in table 2.

Table 2. Module evaluation characteristics

Correct answer	Evaluation type	Normal time	Long time	Min. accuracy
method	Edit distance	15	60	80
Declaration body	Multiple options	10	60	100
constructor	Edit distance	15	60	80
False	Multiple options	10	30	100
{}	Multiple options	10	30	100
Return name	Keywords	15	60	75
new	Keywords	10	30	100
usr = new User();	Code problem	30	300	100

In table 3 we show the results of the student interaction (probabilistic calculations).

Table 3. Probabilistics results for student interaction

Accuracy	Probability
83	0.83
100	0.90
100	1.00
0	0
0	0
75	0.60
100	0.90
94	0.85

As this learning module had assigned a 20% to the practical evaluations (this is designed by the module creator), the probability that this resource has facilitated the learning to the student is of 0.65. This value later is added to the calculations done to other resources of the same type. Thus, at the beginning of another resource, the probabilities can determine that the student belongs to certain characteristics of learning.

In the end, the results of the student evaluation are shown. It is necessary to indicate that the result is different from the one used for calculating the learning type. Also the student will have the possibility of seeing his/her answers and compare them with the correct ones, in addition to receive feedback from the system.

Figure 8 presents the results for the student interaction in this running example.

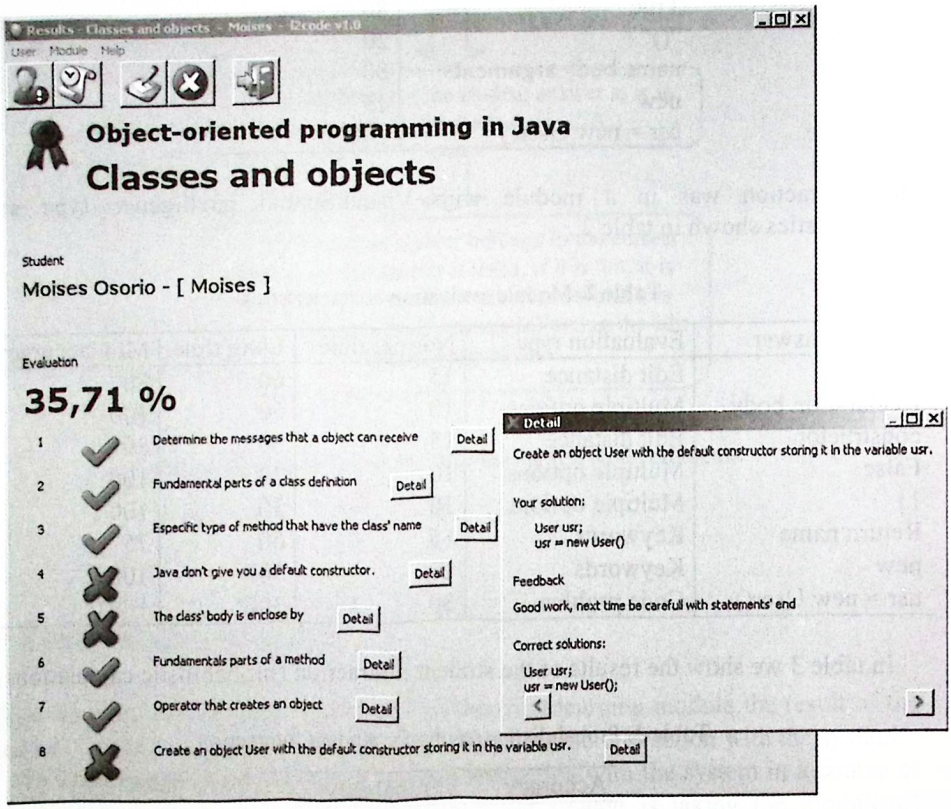

Figure 8. Evaluation results

5 Related Work

A series of research work in this area has been developed for several years; nevertheless all of them are oriented for teaching a single programming language and most of the times for introductory courses. One of them is ITEM/IP [7] an ITS for teaching programming. ITEM/IP is only oriented to provide an introductory course to Turingal (programming language). Another ITS is GREATERP [8] which is based on Anderson's theory of learning and oriented for teaching the LISP language. Last, a system named BITS [9] which is also oriented for teaching only one programming language.

As we can see, one main difference between those other systems with respect to L2Code is that they are oriented to just one programming language. Besides, the exactitude of the student answers depends of just one programming language, whereas L2Code looks for the degree of exactitude of the answer within a set of correct answers defined by the creator of the learning module. That is the main reason why L2Code does not need to specify a programming language or an exact solution but a set of answers that must be validated like correct and another set of answers that must be validated like incorrect allowing identifying the correct feedback to the student.

6 Conclusions

In this paper an ITS named L2Code was described. The system predicts the best learning resources for the students. The learning modules are a set of features that describe when the learning resource must be presented to the student. On entry to a particular unit, the learning scheme predicts which resource the student should use.

Currently some empirical studies are taking place to analyze the reaction of students to the predictive engine in L2Code. This study is examining instructional strategies due to the relationship between them and the learning performance.

Future work involves development of a user-friendly interface to create courses and further analysis in order to identify the relevance of different features. Other work will involve generalizing the adaptive engine to use different categories of learning resources.

References

1. Soloway, E. & Spohrer, J. (1989). Studying the Novice Programmer, Lawrence Erlbaum Associates, Hillsdale, New Jersey. 497 p.
2. Barr, M., Holden, S., Phillips, D. & Greening, T. (1999). An exploration of novice programming errors in an object-oriented environment, SIGCSE Bulletin, 31(4), pp. 42-46.
3. Deek, F., Kimmel, H. & McHugh, J. (1998). Pedagogical changes in the delivery of the first-course in computer science: Problem solving, then programming. Journal of Engineering Education, 87, pp. 313-320.
4. Wiedenbeck, S., Ramalingam, V., Sarasamma, S. & Corritore C. (1999). A comparison of the comprehension of object-oriented and procedural programs by novice programmers. Interacting with Computers, 11(3), pp. 255-282.
5. McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B., Laxer, C., Thomas, L., Utting, I. & Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students, SIGCSE Bulletin, 33(4), pp. 125-180.
6. Edward R. Sykes, Franya Franek: Web-Based Architecture of an Intelligent Tutoring System for Remote Students Learning to Program Java.
7. P. L. Brusilovsky: Intelligent Tutor, Environment and Manual for Introductory Programming

8. Brian Reiser, John Anderson, Robert Farrell: Dynamic Student Modeling in an Intelligent Tutor for LISP Programming
9. C. J. Butz, S. Hua, R. B. Maguire: A Web-Based intelligent Tutoring System for Computer Programming
10. Howard Gardner: Theory of Multiple Intelligences.
11. Declan Kelly, Brendan Tangney: Predicting Learning Characteristics in a Multiple Intelligence Based Tutoring System.
12. Markus Lang: Implementation of Naïve Bayesian Classifiers in Java.
13. Adolfo Duran, Ana Cavalcanti, Augusto Sampaio: Formal Methods and Software Enginnering
14. Robert Cecil Martin: UML for Java Programmers
15. Gosling, Joy, Steele, Bracha: The Java™ Language Specification
16. Jason Hunter, Brett McLaughlin: JDOM™ Project, http://www.jdom.org
17. Eclipse Foundation: SWT (Standard Widget Toolkit), http://www.eclipse.org/swt

Intelligent Biomedical Applications