L2Code: An Author Environment for Hybrid and Personalized Programming Learning

Ramon Zatarain-Cabada, M. L. Barrón-Estrada, J. Moisés Osorio-Velásquez, L. Zepeda-Sánchez, and Carlos A. Reyes-García *

Instituto Tecnológico de Culiacán, Juan de Dios Bátiz s/n Col. Guadalupe, C.P. 88220
Culiacán, México, tel. +52 667 7131796
rzatarain@itculiacan.edu.mx
* Instituto Nacional de Astrofísica, 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 under a hybrid or blinded environment. It was designed and implemented to work with diverse types of modules oriented to certain ways of learning using principles of Multiple Intelligences. The author tool facilitates the creation of adaptive or personalized learning material to be used in multiple-paradigm programming language courses applying an artificial intelligence approach. The Tutoring System works with a predictive engine that uses a Naive Bayes classifier which operates in real time with the knowledge of the historical performance of the student. We show results of the tool.

1 Introduction

Teaching and Learning a programming language is in general considered a tough job, and programming courses usually have high abandon rates. Research has proven that for a beginner to become an expert programmer he might spend more than 10 years [1]. A great amount of 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]. Some research show the difficulties of Object oriented programming by performing a web-based survey for both students and teachers [5].

Our proposal is an Intelligent Tutoring System (ITS) designed to accept diverse types of programming language paradigms oriented to different ways of teaching and learning like e-learning and classroom learning and by using the principles of Multiple Intelligences [6]. This system, named L2Code, can dynamically identify the learning characteristics of the student [7] and provide him personalized material according to his type of intelligence. The different programming modules can be conveniently produced by any instructor. 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 [8] which operates in real time with the knowledge of the historical performance of the student.

The organization 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. Test and 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.

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.

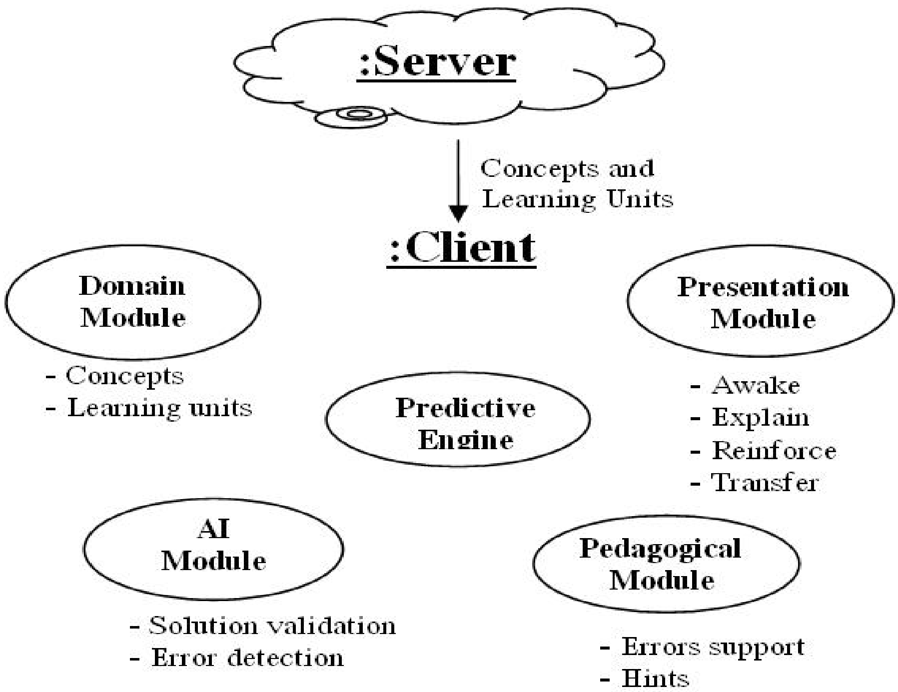

Fig. 1. General architecture of L2Code

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 Learning Process in L2Code

The learning process in a module starts by describing basic information like *name*, *objectives*, *previous* and *further knowledge* of the module. Next, the 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.

2.2 Predictive Engine

As we defined previously, the predictive engine is the one in charge to compute 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 formed 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 correct 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: *Fast, Normal*, and *Long*.
- 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; *No* otherwise.

- 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:
- 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 what 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 deal with 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 [9, 10]. The system was implemented with Java™ [11]. L2Code makes use of two external packages that are: JDOM [12] for the XML reading and writing and SWT (Standard Widget Toolkit) [13] for the creation of native graphical interfaces.

3.1 Naive Bayes Classifier Algorithm

This algorithm (Figure 2) is in charge of the probabilistic computations for making prediction of the right student learning resource. During the interaction of the student with the learning module the attributes of this interaction are recorded and, when finishing it, the corresponding probability of the actual learning resource is updated.

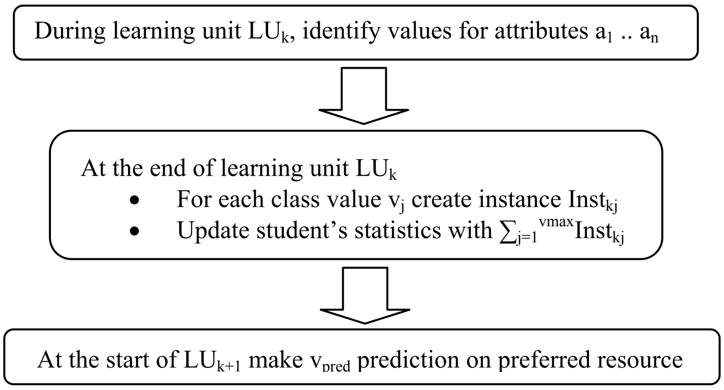

Fig. 2. Naïve Bayes classifier algorithm

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 3.

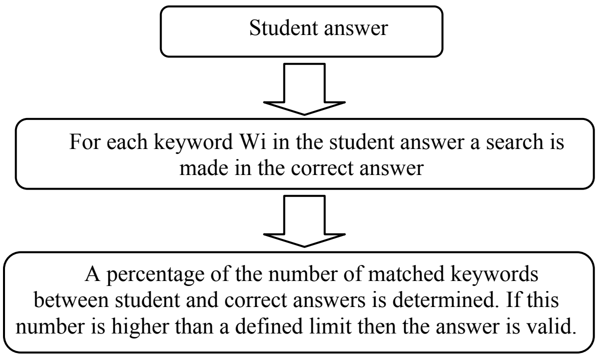

Fig. 3. Evaluation with keywords

- **Edit Distance**. It allows also 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 explained in Figure 4.

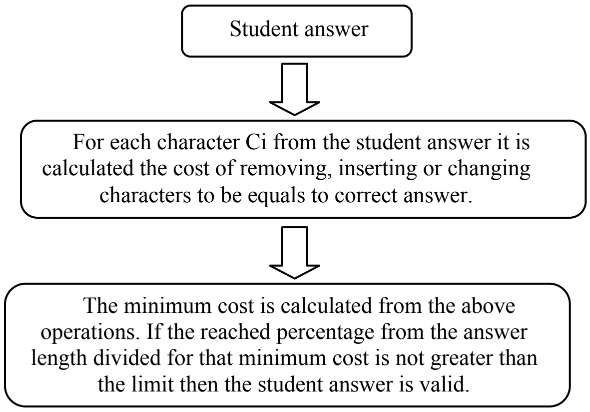

Fig. 4. Evaluation with edit distance algorithm

- **Practice Evaluation (Code Problem).** This type of evaluation (see Figure 5) was implemented to evaluate code and to provide hints to the student throughout its development and, at the end, a feedback of its answer is returned.

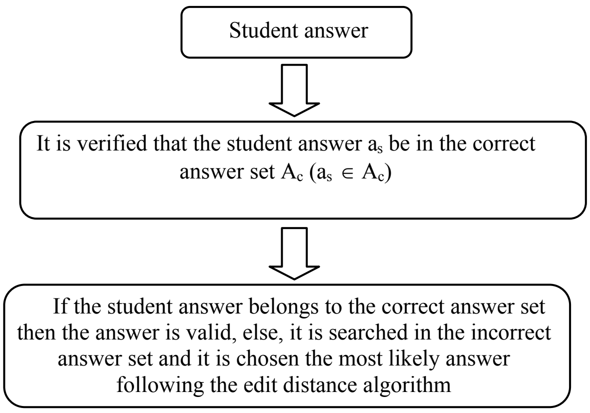

Fig. 5. Algorithm for practice evaluation (code problem)

4 Experimental Results

We will present an example for an object-oriented programming (OOP) course. This course is offered in the computer engineering program of our institution (Instituto Tecnológico de Culiacán). Figure 6 shows the interface of one of the topics. We can observe on the left bottom side of the figure, when the system makes a prediction of the learning style of the student (visual/spatial). We also observe at the right bottom side, the different learning styles the student can choose.

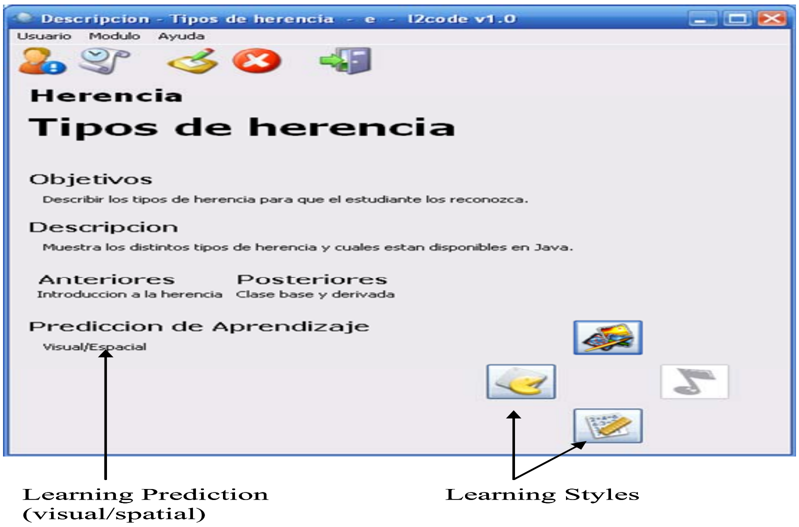

Fig. 6. Choosing the Learning Style

Topic content of multiple inheritances and topic assessment with results are shown in figures 7 and 8.

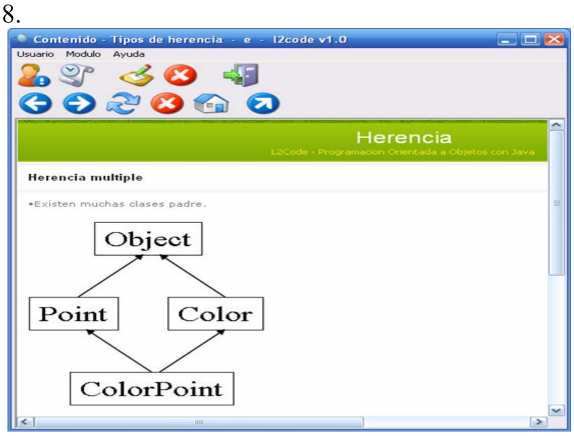

Fig. 7. Course Topic Content

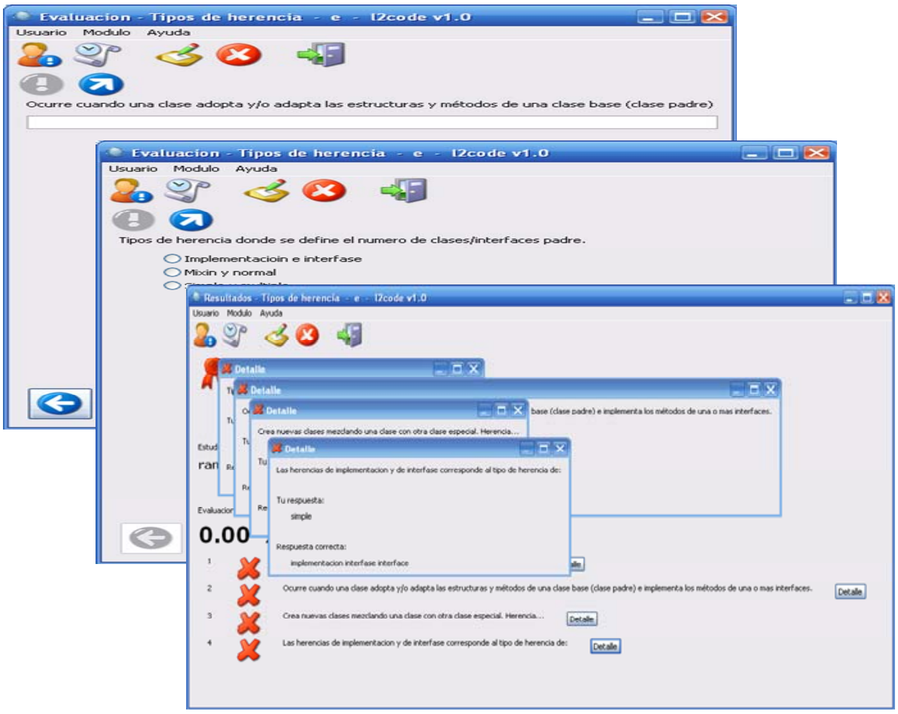

Fig. 8. Interfaces for Topic Assessments and Results

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 shows a student interaction with L2Code. The interaction was in a module with Visual/Spatial intelligence type and the characteristics are shown in Table 2.

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

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 computations).

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 last part, 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.

5 Related Work

Research in this area has been oriented for teaching single programming languages and most of the time for introductory courses. ITEM/IP [14] is an ITS for teaching programming. ITEM/IP is only oriented to provide an introductory course to Turingal (a programming language). GREATERP [15] is another ITS based on Anderson's theory of learning and oriented for teaching the LISP programming language. A system named BITS [16] is also oriented for teaching only one programming language. One disadvantage of those systems is that they are oriented to just one programming language.

6 Conclusions

L2Code predicts the best learning resources and style for the students. The learning modules are a set of features that describe when the learning resource must be presented to the student. When starting any particular unit, the predictive engine calculates which resource the student must use for his learning process.

At present some empirical studies are taking place to analyze the reaction of students to the Object-Oriented Programming Course produced with L2Code. The course combines e-learning and classroom material. This study is examining instructional strategies due to the relationship between them and the learning performance.

Future work involves more implementation development of a user-friendly interface to create courses and further analysis in order to identify the relevance of different features. Also, we are working with other machine learning techniques.

References

1. Soloway, E. & Spohrer, J. *Studying the Novice Programmer*, Lawrence Erlbaum Associates, Hillsdale, New Jersey. 497 p., ISBN: 0805800026, 1988.
2. Barr, M., Holden, S., Phillips, D. & Greening, T. “An exploration of novice programming errors in an object-oriented environment”, SIGCSE Bulletin, 31(4), pp. 42-46, 1999.
3. Deek, F., Kimmel, H. & McHugh, J. “Pedagogical changes in the delivery of the first-course in computer science: Problem solving”, The Programming. Journal of Engineering Education, 87, pp. 313-320, 1998.
4. Wiedenbeck, S., Ramalingam, V., Sarasamma, S. & Corritore C. “A comparison of the comprehension of object-oriented and procedural programs by novice programmers”, Interacting with Computers, 11(3), pp. 255-282, 1999.
5. McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B., Laxer, C., Thomas, L., Utting, I. & Wilusz, T. “A multi-national, multi-institutional study of assessment of programming skills of first-year CS students”, SIGCSE Bulletin, 33(4), pp. 125-180, 2001.
6. Gardner H. Frames of Mind: The theory of multiple intelligences. New York. Basic Books, 1983.
7. Declan Kelly, Brendan Tangney: “Predicting Learning Characteristics in a Multiple Intelligence Based Tutoring System”, Proceedings of the Seventh International Conference on ITS (ITS 04), Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2004.
8. Markus Lang: “Implementation of Naïve Bayesian Classifiers in Java”, http://www.iit.edu/~ipro356f03/ipro/documents/naive-bayes.doc.
9. Ian Sommerville. Software Engineering, Addison-Wesley, ISBN 020139815X, 2001.
10. Robert Cecil Martin: UML for Java Programmers, http://books-support.softbank.co.jp/isbn/pdf/2513.pdf.
11. Gosling, Joy, Steele, Bracha: The Java™ Language Specification.
12. Jason Hunter, Brett McLaughlin: JDOM™ Project, http://www.jdom.org.
13. Eclipse Foundation: SWT (Standard Widget Toolkit), http://www.eclipse.org/swt.

14. P. L. Brusilovsky. *Intelligent Tutor, Environment and Manual for Introductory Programming*, Innovations in Education and Teaching International, Volume 29, Issue 1, pages 26 – 34, 1992.
15. Brian Reiser, John Anderson, Robert Farrell: "Dynamic Student Modeling in an Intelligent Tutor for LISP Programming", IJCAI, pages 8-14, 1985.
16. C. J. Butz, S. Hua, R. B. Maguire. "A Web-Based intelligent Tutoring System for Computer Programming", Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence, p.159-165, 2004.