Evolving technology to better support teaching introductory programming inside Moodle

Leônidas de Oliveira Brandão∗, Igor Moreira Félix∗, Patricia Alves Pereira† and Anarosa Alves Franco Brandão§

∗Instituto de Matemática e Estatística - IME
University of São Paulo
São Paulo, Brazil
Email: leo@ime.usp.br, igormf@ime.usp.br
† Ibmece São Paulo
São Paulo, Brazil
Email: patricia.pereira@ibmece.edu.br
§Escola Politécnica
University of São Paulo
São Paulo, Brazil
Email: anarosa.brandao@poli.usp.br

Abstract—Nowadays, it is increasing the importance of developing skills related to the computational thinking at earlier stages of education, and the adoption of tools that implement the visual programming paradigm has been well succeeded in presenting introductory notions of programming from kindergarten to college. Such tools allows the user to program while manipulating blocks that represent programming languages instructions. iVProg is among this class of tools and was firstly implemented in Java and could be integrated to Moodle using iAssign. Due to technological evolution, iVProg also evolved to iPrgH and its integration to Moodle is still possible by using the new version of iAssign that allow the integration of HTML 5 packages to Moodle. In this paper we describe how this evolution takes place and presents iPrgH functionalities as well as iAssign extension to include HTML 5 packages into Moodle. In addition, we give some teaching support whenever integrated using iPrgH with Moodle using iAssign.

I. INTRODUCTION

The increasing role of Information and Communication Technologies in our daily lives as well as the pervasiveness of computing in several branches of activities makes computational thinking a strong desirable skill. Such a need brings the inclusion of introductory programming disciplines in different education levels. In fact, the discipline of introductory programming is part of other curriculum than the STEM one, but also of humanities and medicine. In addition, it was also included in elementary and high school [1, 2].

Computational thinking involves the ability of representing and solving a problem in a way that could be “easily” translated in a programming language [3, 4]. This is why introducing the first notions of programming at earlier stages of education is so important.

Nevertheless, for those who were at college or universities at the first decade of the 2000, that are numbers to attest the fail rate in introductory programming courses was about 33% [5, 6, 7]. Bosse and Gerosa [8] report that more than 50% of students enrolled in MAC115, an introductory course of programming at USP, during the years of 2010 to 2014, had failed.

Although the numbers are recent, this is not a novelty: Du Boulay [9] stated in 1986 that “Learning to program is not easy”. Moreover, there is an awareness that the lack of skills and cognitive abilities to formulate and solve problems algorithmically within novices result in high fail and drop out rates.

In this context, the proposition of approaches to facilitate the acquisition of such skills and cognitive abilities is strongly desirable. It is worth to mention that acquiring such skills and abilities is beyond learning an specific programming language like C, Java or Phyton. The visual programming paradigm is one of the approaches adopted to teach and learn algorithms, which are computational solutions for problems or, in other words, description of computational thinking. We can cite, among existing visual programming tools, Alice [10], Greenfoot [11], Scratch [12] and iVProg [13].

By adopting visual programming to teach introductory programming, we use graphical components to allow the
construction of algorithms while simply manipulating them and reducing the cognitive load related to the syntax of traditional programming languages [10, 11, 14, 15, 16]. Moreover, due to its nature of stimulating computational thinking, tools to support visual programming has been successfully used even in advanced programming courses, such as distributed programming [16].

Whenever we think about introducing algorithms for novices at universities and colleges, it is of particular interest that tools to support the teaching process could be available anywhere, anytime, integrated to web-based learning environments, such as Moodle.

In this paper we present $iVProgH$, the HTML 51 version of $iVProg$, and an extension of $iAssign$ [17] to incorporate interactive Learning Modules (iLM) [18] coded in HTML 5 into Moodle. These are technological evolution of existing tools to address new constraints of web browsers.

The paper is structured as follows: section II briefly presents $iVProg$ and $iAssign$ in their earlier versions, section III presents $iVProgH$ and its model to perform automatic assessment, section IV presents the extension of $iAssign$ to allow the integration of interactive Learning Modules (iLM) coded in HTML 5 in Moodle. In addition, section V presents some guidelines to use $iVProgH$ integrated to Moodle and, finally, in section VI we present our final considerations and conclusion.

II. $iVProg$ AND $iAssign$

In this section we present two of the $LInE^{2}$ free educational systems, the $iVProg$ and the $iAssign$. The first one is an educational software to be used to present the concepts of programming, and the second is a software to integrate other educational Web software to Moodle, when these software could be classified as interactive Learning Modules (iLM) [18, 19, 20, 21]. Besides, all $iVProg$ versions can be considered iLM, so they can be integrated to Moodle by the $iAssign$.

A. $iVProg$

$iVProg^{3}$ (interactive Visual Programming in the internet) is a tool to support teaching and learning of introductory programming. Its first version was launched in 2009 [14] and a refactored version is available since 2015 [22]. The conception of $iVProg$ was guided by the following requirements:

• allow its integration to a web-based system or platform to support teaching and learning;
• allow the authoring of exercises to be deployed in repositories;
• provide automatic evaluation of exercises;
• decrease the cognitive load of learning the syntax of an specific programming language while learning algorithms.

1This means HTML (Hypertext Markup Language) version 5, enriched by JavaScript and CSS (Cascading Style Sheets).
2Laboratory of Informatics in Education, sited on the University of São Paulo - http://line.ime.usp.br.
3$iVProg$: available at http://www.matematica.br/ivprog/

The first version of $iVProg$ was a simplification of Alice 2.0 [10]. By simplification we mean extracting some functionalities, such as 3D animation, to allow its use as an applet4. A screenshot of its interface is presented in figure 1, with an algorithm to calculate $n!$ (the factorial of a non negative integer $n$). This version could be executed within a web browser (applet) or individually, as a Java application.

![Figure 1. $iVProg$ (Java) - version 1](image1)

The second version of $iVProg$ was a refactored version of the first one. In fact, it was conceived as a product of an iLM software product line [21]. The interface of this

4mini Java applications that had used to run in any web browser.
version is presented in figure 2, with the same example presented in figure 1. Its interface includes the commands in a contextualized way, differently from its first version and Alice. Since it was first launched, iVProg has been successfully used by teachers and students in introductory programming courses [13], most of the time integrated with Moodle using iAssign. However, since 2015 some web browsers started experiencing problems to run applets. This was because it depends on the NPAPI (Netscape Plug-in API) technology to properly run and, for instance, Firefox/Mozilla does not include NPAPI since version 52, Chrome since version 45.

For this reason we started a new iVProg version, now implemented in HTML 5, and initiated the extension of iAssign to allow the incorporation of iLM coded in HTML 5 to Moodle. The iVProg in HTML 5 is presented in section III and the extension of iAssign is presented in section V.

B. iAssign

iAssign is one of the available modules to extend the Moodle system. It brought to Moodle the main idea introduced by SAW (Web Learning System) [18, 19, 20] package with resources for improving its interactivity by allowing the integration of iLM into educational contexts offered by the system. The main functionalities provided by iAssign are: interactive activity, iLM filter, and detailed report [17]. A screenshot of the iAssign interface is given in figure 3.

Figure 3. iAssign activities’ main interface

An Interactive Activity is a resource for authoring interactive activities (exercises or examples) using iLM; an iLM filter is a resource for integrating iLM to Moodle, making possible to use the iLM in any Moodle’s asynchronous context, such as forum, glossary, wiki etc.

Figure 4 shows the use of an iLM to support teaching Geometry in the context of a glossary. At the top of figure 4 we have the authoring interface of the Moodle’s glossary activity. In order to include the iLM to turn the activity into an interactive activity it is needed to insert the iLM file name between specific tags delimiters (<ia>···</ia>) where it must be displayed within the text. The bottom of figure 4 shows the way the interactive activity is displayed for students.

A detailed report provides information related to the performance of students while doing the interactive activities.

It includes information about how many attempts the student got to finish the activity and, if the iLM provides automatic assessment resources, it presents the status of the activity, e.g, if it was evaluated as correct or wrong (see figure 5).

Figure 4. iLM filter - including an Interactive Activity of Geometry.

Figure 5. Detailed report of activities (green tick means correct; red wrong)

III. iVProgH

iVProgH is an evolving implementation of iVProg version 2, now in HTML 5, to provide interoperability among all current Web browsers. Figure 6 presents its main interface, which is very similar to the previous version (see figure 2). In addition, iVProgH can be easily incorporated to the any web-based environment by simply writing a few lines of HTML code. iVProgH maintains the commands contextualized, as it was in iVProg version 2. This model brings as advantage to learners the reduction of options available in the graphical interface. For example, in an initial context, any command is available (see left block in figure 7), but after creating a selection command, the options available in the context of selection are automatically restricted to the viable ones, which mean, the creation of logical expressions (see the right side in figure 7).

In the iVProgH stable version, the following types of variables are available: integer, real, text and boolean. In addition, it provides resources to attributions, two variations of loops (while and for), commands for control flow (logical
selection) and command for input/output (i.e., read and print). Comparing both versions, iVProgH innovates in the authoring interface provided to teachers, to create interactive activities with automatic assessment.

Automated assessment in iVProgH is implemented based on test cases. Therefore, in order to prepare an exercise to have automatic assessment, the teacher must create a list of inputs and their corresponding outputs, the expected outputs. The expected outputs are used to compare the values provided by the one who solve the exercise and press the bottom to send it for evaluation.

1) Test Case design: The aim of test cases is to simulate the behavior of a correct algorithm. For example, in an activity targeting the output of the average of two integer values (entered by the user), one test case could be: using as inputs the list \((-1, 102)\) and as its corresponding output the single value 51.5 (i.e. \((-1 + 102)/2 = 101/2\)). A second test could have as input the values \(-1\) and 101, and the correspondent output 50.

In formal terms, each test case consists of two lists, the input list \((e_1, e_2, \ldots, e_k)\) and its correspondent (if using a correct algorithm) output list \((s_1,1, s_1,2, \ldots, s_1,l_1, s_2,1, s_2,2, \ldots, s_2,l_2, \ldots s_k,1, s_k,2, \ldots, s_k,l_k)\). On one hand, when the student’s algorithm is tested, iVProgH uses \(e_1\) as its first input, \(e_2\) as the second input, and so on. On the other hand, if \((o_1,1, o_1,2, \ldots, o_1,l_1, o_2,1, o_2,2, \ldots, o_l,1, o_k,1, o_k,2, \ldots, o_k,l_k)\) are the outputs from the student’s algorithm when using the input list, these are compared with the expected output (some distance metric like \(\sum_{i=1}^{k} \sum_{j=1}^{l_i} ||o_{i,j} - s_{i,j}||\)).

To avoid mistakes, it is recommended that the teacher implements a complete version of the expected algorithm to create the list of inputs and expected results to the activity. Having the algorithm, the teacher must provides the most significant inputs to the problem and submits them to the it. The output list will be formed by the algorithm’s outputs, in the very same order. The process of generating a list of test cases is represented in figure 8.

It’s possible to illustrate the use of test cases using a simple problem: Develop an algorithm that reads an integer value and prints this number and its square. To this problem, the test cases could be the following six pair of lists (table I):

2) Solving an exercise in iVProgH: creating and algorithm: The students must have access to the problem statement, devise a solution for it, implement an algorithm in iVprog (that solves the problem) and then send it to evaluation. This process is represented in figure 9.

When the student finishes one version of algorithm-solution and “press” evaluation button, the iVProg proceeds to the following steps: 1. gets the first item \(e_1\), from the first test case, and submits it to student’s algorithm; 2. captures the outputs generated by the input \((o_1,1, o_1,2, \ldots, o_1,l_1)\); and 3. compares them to the expected outputs \((s_1,1, s_1,2, \ldots, s_1,l_1)\).

Depends on the iLM configuration, it is possible to evaluate if the algorithm is correct without sending it to the server, as many times as the student wish. In current version of iAssign, only the last student’s submission (and its grade) is registered.

3) Automatic Assessment in iVProgH: As aforementioned, iVProgH compares the outputs sent after the solution of an exercise with the expected ones, resulting in a grade between 0 and 1. If there is no difference between the expected output and the sent output, it is attributed integral grade 1. However if significant difference is detected in the sent outputs, the grade should be zero (0). iVProgH repeats this process for all entries
and, at the end of the list of test cases, it generates a mean of grades.

The assessment process is presented in diagram at the figure 10. The algorithm is executed using the input list \( (e_1, \ldots, e_k) \). Then the sent outputs \( (o_{1,1}, \ldots, o_{k,l}) \) are compared with the expected outputs \( (s_{1,1}, \ldots, s_{k,l}) \). The automatic assessment process begins with the reception of the algorithm sent for evaluation, then it is executed for the whole list of test cases inputs. Finally, the comparison among the generated outputs and the expected results and its associate results are presented (“printed”) in the iVProgH output area.

![Algorithm Evaluation Process](image)

Figure 10. Algorithm Evaluation Process

It’s important to highlight that the test case evaluation’s efficiency is directly related to their coverage extension. This mean that teachers must provide test cases for each possible variations for the algorithm behavior. For instance, if the purpose of an algorithm is to determine the highest value in a sequence, it is necessary that at least one test case includes the highest number in the first position and other test must have the highest number at the end of the sequence.

IV. INTEGRATOR OF iLM TO Moodle via iAssign

Since the first version of iAssign, in 2010, it provides the integration of Java packages to the Moodle environment. This means that any Moodle administrator can install new iLM, teachers can produce new interactive activities to their students with this iLM, and students can view, test and send their solutions, in an integrated fashion. Besides, iAssign has usual features to import, export, duplicate, remove, and configure. In the figure 11 is presented a components diagram for Moodle-iAssign-iLM. In the diagram we can devise that iAssign is integrated with Moodle by its Activity modules and Filters.

![Components of Moodle-iAssign-iLM](image)

To allow the incorporation of HTML 5, new PHP scripts were developed and others were re-factored, resulting in 32 new or modified scripts, from a total of 51. The current directory structure is presented in table II.

<table>
<thead>
<tr>
<th>Directories</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>mod/iassign</td>
<td>iAssign directory inside Moodle</td>
</tr>
<tr>
<td>mod/iassign/backup/moodle2</td>
<td>containing files to provide backups/restoration</td>
</tr>
<tr>
<td>mod/iassign/classes/event</td>
<td>treatment to event</td>
</tr>
<tr>
<td>mod/iassign/db</td>
<td>database management (updates, upgrades, …)</td>
</tr>
<tr>
<td>mod/iassign/icon</td>
<td>support imagens</td>
</tr>
<tr>
<td>mod/iassign/ilm</td>
<td>directory to all the installed iLM</td>
</tr>
<tr>
<td>mod/iassign/ilm_debug</td>
<td>to help debug during development</td>
</tr>
<tr>
<td>mod/iassign/ilm_handlers</td>
<td>new directory to manage iLM Java and HTML 5</td>
</tr>
<tr>
<td>mod/iassign/lang</td>
<td>language dictionary (PT_BR, EN_US, FR)</td>
</tr>
<tr>
<td>mod/iassign/pix</td>
<td>to provide iAssign logo</td>
</tr>
</tbody>
</table>

To be integrated into Moodle, an HTML 5 package, as well as any other iLM, has to implement at least the first two methods described below:

- a method to read a content file through URL (so iAssign can inform the parameter iLM_PARAM_Assignment to the iLM package);
- a method named getAnswer(), to returns to iAssign a text with the student’s answer to the activity; and
- if the iLM contains an engine to automatic assessment, a method named getEvaluation().

If the iLM implements automatic assessment, like iVProgH, iAssign manages the results obtained by student in the activity, inserting the activity’s grade in the Moodle gradebook.

For security reasons, the integration of new iLM is restricted to users with administrative permissions in Moodle system. To add a new instance or a new version of a previously installed iLM, the user accesses the administrative area, enables block edition, selects the tab plugins and, in the block activities, selects the package iAssign. Figure 12 shows the Moodle interface for such installation.

![Adding a new iLM in Moodle via iAssign](image)

Figure 12. Adding a new iLM in Moodle via iAssign

When the new iLM has been successfully installed, users with teacher’s role in some course, are able to create new activities using the recent added iLM and their students may solve such activities.
A. iLM integrated to Moodle: operation

When integrated into Moodle, any available iLM is ready to be used. The teacher can author interactive activities using the iLM and make them available to the students. In their turn, the students can solve proposed problems using the iLM, test their solution (if the iLM provides automatic assessment mechanism), and send it to the server. The teacher can see the student’s answer. If the iLM has a mechanism to automatic assessment, then iAssign also manages the associated grades, integrating them to the Moodle’s gradebook (or Grader report).

It is worth note that, iAssign considers as grade a value between 0 and 1 (0.5 means medium grade) that is registered in the Moodle database.

V. iVProgH integrated to Moodle via iAssign

Any iLM can be considered as a coarse-grained learning object (LO) since they can be found, reused, and are interoperable [23, 24]. Besides, the essence of LO is to be devoted to education and be used via Web. Several Learning Management System (LMS), like Moodle, provides methods to allow the LO integration and usage.

In this sense, the iAssign is a Moodle integrator of iLM, like iVProgH. However its use is tight integrated, as explained in previous sections. Moreover, since iVProgH provides automatic assessment, the teacher can easily detect conceptual flaws or, even, if an activity is not well adjusted (maybe presenting some conception error or the difficulty is over the students knowledge). For instance, in a period of time, if no student could solved the problem, this could mean that the statement is ambiguous or contains errors.

To the students, a significant advantage of iVProgH-iAssign over a weak integrated LO could be the facility to use and detect problems in their solution. In the following, we will describe how an activity can be authored using iVProgH-iAssign, as well as how this activity can be executed. A sequence diagram of this operation is given in Figure 13.

1) Activity authoring - teacher role: The iVProgH provides an interface for teachers to create activities, actually to create test cases, as explained in section III. The initial step is the common one to any Moodle activity creation (turning on the edition mode; selecting the Add an activity or resource and iAssign option). After the iAssign block of activity is created (as in figure 3), it must be selected inside it the option Add activity and chosen the iVProgH among the iLM options. In this context, it must be opened the online editor, that will result in the test case builder in figure 14, the left rectangle.

![Figure 14. Activity authoring](image)

2) Working out on an activity - student role: The activity is available to the student in a Web page with the iLM, and the iLM loads the corresponding activity file. In the iVProg example, the file with its test cases.

Using as example the problem: Build an algorithm that, after reading an integer number, it prints the square of this number, an example of solution is presented in figure 15.

If the student wants, it is possible to run the algorithm, by “clicking” the run button (represented as a black triangle in the figure 15). The output is presented at the console area (the gray area, at the bottom of the figure).

VI. Final Considerations

In this paper we presented the evolution of a suite of tools to support the teaching and learning algorithms and introductory programming in web-based learning environments such as Moodle.

The suite of tools is composed of an iLM and a Moodle package for integrating iLMs to Moodle. The iLM is a visual
programming tool with automatic assessment resources, the iVPrgH, and the integrator is iAssign. This suite evolution aims to provide interoperability among all existing web browsers and avoid problems with incompatibility concerning legacy Java applets. Therefore, iVPrgH is implemented in HTML 5 and iAssign was extended to integrate HTML 5 modules to Moodle.

This evolution will be extended to all the available iLM in order to support interactive learning of issues like Geometry, Calculus, and Combinatorics.

REFERENCES


