PETCHA
A Programming Exercises Teaching Assistant

ABSTRACT
This paper presents a tool called Petcha that acts as an automated Teaching Assistant in computer programming courses. The ultimate objective of Petcha is to increase the number of programming exercises effectively solved by students. Petcha meets this objective by helping both teachers to author programming exercises and students to solve them. It also coordinates a network of heterogeneous systems, integrating automatic program evaluators, learning management systems, learning object repositories and integrated programming environments. This paper presents the concept and the design of Petcha and sets this tool in a service oriented architecture for managing learning processes based on the automatic evaluation of programming exercises. The paper presents also a case study that validates the use of Petcha and of the proposed architecture.

Categories and Subject Descriptors
D.3.3 [**Programming Languages**]: K.3.1 [Computer Uses in Education]: Computer-assisted instruction (CAI), Distance learning; K.3.2 [Computer and Information Science Education]: Computer science education, Self-assessment.

General Terms
Design, Experimentation, Standardization, Languages.

Keywords
Teaching Assistant, Automatic Evaluation, Programming Exercises, Interoperability, Learning Objects.

1. INTRODUCTION
A teaching assistant (TA) is a person who assists a teacher, typically in practical classes. The task of a TA in a programming course is usually to help students in solving exercises and assignments. They must help students to use programming tools (integrated programming environments, compilers, and debuggers), check if they have solved the exercises and provide feedback to help them to overcome their difficulties. Unfortunately, the number of TAs is frequently insufficient for the number of students enrolled in introductory programming courses, and they are only available on computer labs and on a certain timetable.

This research aims to create an automatic TA specialized in programming exercises, a tool for bridging between the teacher and the students, and providing them with the best systems for each task. The automatic TA presented in this paper is called *Petcha*, an acronym of Programming Exercises TeaCHing Assistant.

Petcha can be described as a *scaffolding* tool since it complements existing tools and was designed to be easily removed once it is no longer needed. For instance, rather than providing its own environment for solving exercises Petcha promotes the use of existing Integrated Development Environments (IDEs) and different IDEs can be can be used with Petcha, such as Eclipse or Visual Studio.

More than just a scaffolding tool, Petcha is also pivot component on a network integrating other e-Learning systems. These e-Learning systems are used for: 1) automatic evaluation of programs and feedback generation; 2) authoring and storing of programming exercises as learning objects; 3) managing instruction and learning activities. A proper integration of these tools sets up the necessary foundations for the practice of solving programming exercises and has a great impact on the acquisition of programming skills.

The remainder of this paper is organized as follows. Section 2 reviews related work on the authoring of learning objects and evaluation of programming exercises. In the following section we present the design of Petcha and its role in the context of an e-Learning environment with a service-oriented architecture. The paper proceeds with a case study where Petcha was successfully used in the practical classes on an introductory programming course. Finally, we conclude with a summary of the main contributions of this work and a perspective of future research.

2. RELATED WORK
Petcha has two facets: it helps the teacher in exercise authoring and the students in the exercise solving. To the best of authors' knowledge, no other tool described in the literature integrates these two facets. Hence, we present in this section several works regarding these two facets separately. Firstly we introduce systems that automate the exercise authoring. Secondly we present systems for the automatic evaluation of programming exercises.

2.1 Exercises Authoring
In recent years, a large number of programming exercises have been developed and published mostly for use in programming contests. These exercises are generally stored in proprietary systems (e.g. online judges) for their own use (e.g. automatic submission, grading). Despite some efforts [8, 33] to define a common format to describe programming exercises, each of these systems has its own exercise format, hindering its sharing among instructors and students.

Trætteberg [31], presented a specification-based and test-driven exercise support plug-in for Eclipse, named JExercise. The plugin gives feedback to students about his/hers progress. An exercise in JExercise is based on three elements: 1) a textual specification of the Java elements that are required and their desired behavior; 2) a set of JUnit tests for checking whether the student's code meets the specification; 3) a model of the required solution. The exercises are wrapped in a Zip-file containing XML and HTML files describing the exercises and test files for testing the student's code, and may additionally contain Java source files and program skeletons.

Jena [11] presents a work regarding the authoring and sharing of programming exercises as learning objects. The work uses Labrat as the automatic grading system. The assignments are composed by an exercise statement, metadata, program solution and tests. All these resources are described with metadata based on the IEEE LOM specification and packaged in a single zip file structure conforming to the format required by Labrat. The exercises are stored in a repository called CollabX.

2.2 Automatic Evaluation of Exercises

The assessment of programming assignments poses significant demands on the teachers' time and other resources [5]. This demand stimulated the development of automated learning and assessment systems in many universities [1]. Most of these systems provide other features such as multi-programming language support, evaluation type (static or dynamic), feedback, interoperability, learning context, security and plagiarism.

Early efforts [10, 20, 24, 26] only support the assessment of exercises in a single programming language. With the advent of the Internet and the increase of platforms heterogeneity, web interfaces began to play an important role in the dissemination of several systems [3, 12, 13, 21]. These systems also share the common feature of supporting the submissions of exercises written in *multi-programming languages* such as Java, C++ and the C.

Regarding the *evaluation type,* the standard way of evaluating a program is to compile it and then execute it with a set of test cases with input and output files (black-box approach). The program is classified as accepted if compiles without errors and the output of each execution test is the same as the expected output. This strategy has been shown to bring undesirable pedagogical issues such as student frustration and confusion [28, 30]. Several systems [3, 10, 12, 18, 21, 26] test not only the behavior of single programs but also analyze the structure of source code (white-box approach). This approach allows the evaluator systems to guarantee that whether the program have been written in a particular way, following a particular algorithm or even using certain data structures. In the field of the output correctness determination, Edwards [6] and Auffarth [2] use also unit tests defined by teachers to validate students' submissions. Other main issue lies with the non-determinism of the program outputs where different correct (or acceptable) solutions to the same programming exercise may not always produce exactly the same output [29]. Leal [13], deals with this non-determinism using dynamic correctors as special evaluators that are invoked after each test case execution. For instance, if the solution is a set of values that can be presented in any order then a dynamic corrector can be used to reduce the output to a normal form.

Depending of the learning context (competitive or curricular) the systems must provide *feedback* to facilitate the students' comprehension about the correctness of their attempt to solve a particular exercise. The generation of feedback relies on static and dynamic program analyses [1]. The development of automatic program evaluation systems with high quality feedback (e.g. compilation errors, execution errors, execution tests) show good results [9, 17] and along with visual, incremental and personalized feedback should shape the future regarding this topic [27].

The *interoperability* of the evaluation systems is also a main issue. An evaluator should be able to participate in learning scenarios where teachers can create exercises, store them in a repository and reference them in a Learning Management System (LMS) and students solve the exercises and submit to evaluators who delivers an evaluation report back to students. Several systems [16] try to address this issue allowing the integration with course management systems. Nowadays with the advent of service oriented architectures the trend is service orientation rather than component-based systems. An evaluator system as a service will automate the existent business logic in distributed e-Learning scenarios allowing more flexibility in the comprised workflows and keeping the systems simple and easy maintainable. Leal [15] specified a service for programming exercises evaluation in a well-known e-Learning framework called the E-Framework. This work was used in Edujudge [32] with promising results.

Concerning the *security* issue, Luck and Joy [16] analyzed this issue covering robust environments, privacy, and data integrity. Security can be handled from ad-hoc solutions to solutions based on Virtual Machines (VM) to execute the programs on a safe and controlled environment.

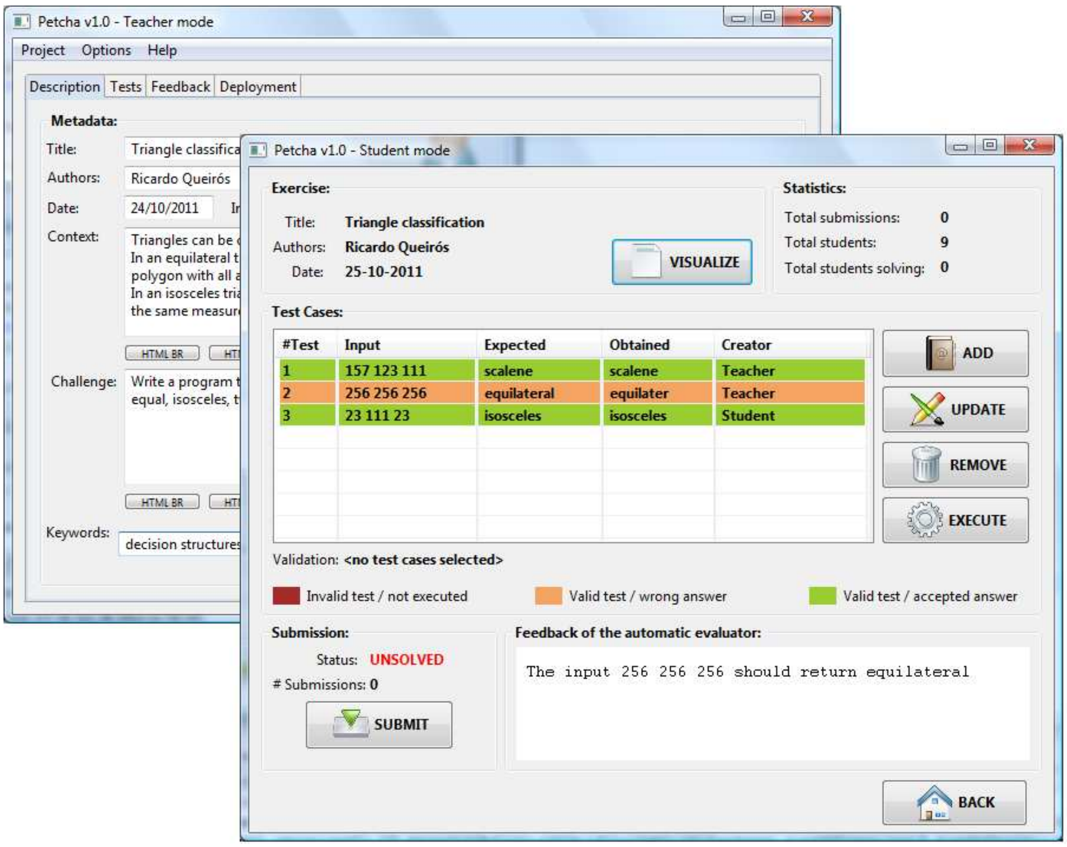
Other concerning is the increase in *plagiarism* [4, 7]. Various systems [3, 16] analyze the integration of plagiarism services in the assessment workflow.

Regarding the *learning context***,** evaluation systems can be used in two contexts: curricular and competitive learning. In the former, teachers use practical classes, assignments and examinations to evaluate students' evolution. The latter relies on the competitiveness of students to increase their programming skills mostly in computer programming contests. In this last context, automated judge systems (or online judges) are used to run programming contests and to practice for such contests. These systems include automatic evaluators and many of these systems organize their own contests such as Mooshak [13], UVA-OJ (University of Valladolid Online Judge), SPOJ (Sphere Online Judge), DOMJudge and others.

3. PETCHA

As happens with a human TA, Petcha needs to interact both with teachers and students. Thus, these two use cases provide an overview of Petcha features. Unlike a human TA Petcha delegates most of its work to others, as it is fundamentally a coordinator of e-Learning systems.

The following subsections present the teacher and student use cases as well as the architecture of a network of e-Learning systems coordinated by Petcha.


Petcha v1.0 - Teacher mode
Project Options Help
Description Tests Feedback Deployment
HTML BR
Petcha v1.0 - Student mode
VISUALIZE
ADD
UPDATE
REMOVE
EXECUTE
SUBMIT
BACK

Metadata:

Title: Triangle classifica

Authors: Ricardo Queirós

Date: 24/10/2011

Context: Triangles can be c
In an equilateral t
polygon with all a
In an isosceles tria
the same measur

Challenge: Write a program t
equal, isosceles, t

Keywords: decision structures

Exercise:

Title: **Triangle classification**

Authors: **Ricardo Queirós**

Date: **25-10-2011**

Statistics:

Total submissions:	0
Total students:	9
Total students solving:	0

Test Cases:

#Test	Input	Expected	Obtained	Creator
1	157 123 111	scalene	scalene	Teacher
2	256 256 256	equilateral	equilater	Teacher
3	23 111 23	isosceles	isosceles	Student

Validation: **<no test cases selected>**

🟥 Invalid test / not executed 🟧 Valid test / wrong answer 🟩 Valid test / accepted answer

Submission:

Status: **UNSOLVED**

# Submissions: **0**

Feedback of the automatic evaluator:

```
The input 256 256 256 should return equilateral
```

Figure 1. The GUI of Petcha with teacher and student modes.

3.1 Use Cases

Petcha is an automatic TA with two main tasks: to assist teachers in the authoring exercises and to help students in solving them. Although complementary, these two tasks share a number of requirements. Both teacher and student need to: code and test programs in an IDE; send and retrieve learning objects from a Learning Objects Repository (LOR); check program code against test cases. Thus, although the graphical user interface of both user profiles shown in Figure 1 is apparently very different, they actually share many Petcha internal functions. The following sub-subsections present both use cases in more detail.

3.1.1 *Teacher*

To author and deploy a programming exercise in Petcha teachers must perform the following three tasks:

Create programming exercises. In the authoring task, teachers automatically create most of the resources related with programming exercises such as expositive resources (e.g. exercise description) and evaluation resources (e.g. test cases, correctors, feedback files). The upper left window of Figure 1 shows an example where the teacher is defining a problem and setting related metadata. Other tabs in this window are used for defining tests, assigning feedback to error patterns and publishing the exercise. All the resources defined in these tabs are encoded using an XML dialect called PExIL (Programming Exercises Interoperability Language) [22]. The aim of PExIL is to consolidate all the data required in the programming exercise utilization, from creation to evaluation, covering also solving, the grading and feedback. The PExIL definition supports the concept of incremental feedback to control the appearance of both types of feedback upon a submission of a student's attempt.

Deploy programming exercises in a repository. In the deployment task, teachers can package and publish programming exercises in repositories. The packaging subtask consists on the selection of a package format and its generation. By default, Petcha supports the IMS Common Cartridge (IMS CC) specification as the package format. An IMS CC learning object assembles resources and metadata into a distribution medium, typically a file archive in ZIP format, with its content described by a manifest file. The generation of an IMS CC package is performed in two steps. First the manifest is generated from a valid PExIL instance and all the resources are assembled in a ZIP package. After that, teachers must publish the package on a repository. In order to be an eligible repository (Petcha uses the crimsonHex repository [14]) one must adhere to content (IMS CC) and communication specifications (IMS Digital Repositories Interoperability – IMS DRI).

Configure programming activity in LMS. For this task teachers search in repositories for suitable programming exercises and store a reference to them in a LMS as a Learning Tools Interoperability (LTI) activity. The LTI specification provides a uniform standards-based extension point in LMSs allowing the integration of remote web tools. Presently, most reference LMSs (Petcha is currently being used with Moodle) do not offer support for the full LTI specification. For this reason, Petcha uses a subset of the LTI specification known as IMS Basic LTI (bLTI). With bLTI a unidirectional link between the LMS and Petcha is created. On the invocation contextual information is provided to the launched process such as user identity, course information and role information. The full support of this specification will allow the access to run-time services on the LMS, enabling Petcha to send evaluation results back to the LMS grade book, for instance.

3.1.2 Student

To solve programming exercises using Petcha students performs the following two tasks:

Select an activity in the LMS. In this task, students should select the activity defined by the teacher in the LMS. This selection triggers an LTI launch of Petcha. The launch includes student's contextual information that can be used to presentation purposes (e.g. personalize the Petcha frontend) or for sequencing purposes (e.g. assign an exercises sequence model). After the selection of the activity Petcha is launched as a Java Web Start (JAWS) application on the computer of the student. This approach enables the interaction of Petcha with the IDE by using shell commands.

Execute the activity using the IDE and Petcha. When a student starts solving a problem Petcha automatically creates a project on the IDE of the student. Currently Petcha supports two IDEs: Eclipse and Visual Studio Express. Other IDEs could be used by extending Petcha's code. Then the student reads the exercise description in Petcha's GUI and solves it on the IDE. The student should test the code locally by executing the teachers' test cases and is encouraged to create new ones. If new test cases are created, a validation step is performed to verify that they meet the specification defined by the teacher in the authoring phase. The right window on Figure 1 shows an example where the student's code did not pass all the local tests (two provided by the teacher and one new test created by the student). Even so, the student decided to submit the code to the evaluator and received a feedback message indicating an input data that generated a wrong answer. After testing, the student should submit the solution to the Evaluation Engine (EE) where the submission is checked against the complete test set provided by the teacher. The report on the evaluation returned by the EE is presented to the student. The student may submit repeatedly, integrating the feedback received from the EE. In the end of this cycle, Petcha reports the exercise usage data back to the repository.

3.2 Architecture

In this subsection we present the overall architecture of a network of e-Learning systems participating in a network coordinated by Petcha. In this network Petcha acts as a pivot component mediating the communication among all components. The architecture depicted by UML component diagram in Figure 2 is composed by the following systems and tools:

Learning Objects Repository to store/retrieve exercises;
Evaluation Engine to evaluate students' exercises;
Learning Management System to present the exercises to students;
Integrated Development Environment to code the exercises.

Petcha coordinates the communication among all the components of the network, from the LMS where students receives the activity to the IDE where students solve them. In order to fulfill this goal, the integration of the pivot component with the other systems must rely on content and communication standards. Using content and communication standards we can abstract the use of specific systems for each type of system. For instance, we can use on this network any repository as long it supports the IMS CC specification to formalize the description of programming exercises and it implements the IMS DRI specification for communication with other services.

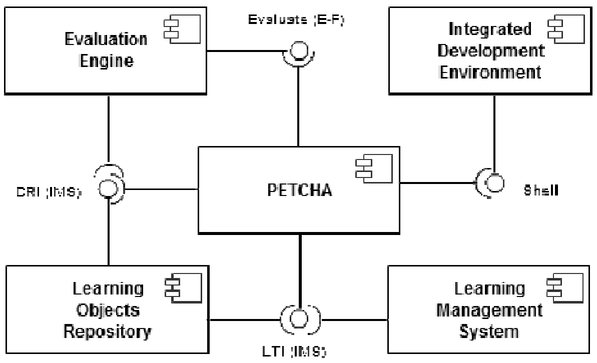

Evaluation Engine
Evaluate (E-F)
Integrated Development Environment
DRI (IMS)
PETCHA
Shell
Learning Objects Repository
LTI (IMS)
Learning Management System

Figure 2. Network component diagram.

Another important point was the choice of the systems that comprise the current network. Since we made several efforts to address interoperability issues, the selection of the tools was straightforward. On the LMS side we choose Moodle since it is a popular and open source LMS, arguably the most popular LMS nowadays. We used the version 1.9 that supports the Basic LTI specification with the further installation of an IMS bLTI consumer[1]. Currently, the version 2.2 supports the IMS LTI 1.1 (a merge version of basic and full LTI) and import IMS CC packages. The exportation of CC packages will come in version 2.3. We successfully tested also the Sakai LMS on this network evidencing the interoperable characteristics of the proposed approach.

For the LOR selection, we had more difficulties to find a system that supports the defined content and communication specifications respectively the IMS CC and IMS DRI specifications. The final choice fell on a home-made system called CrimsonHex - a repository of programming exercises described as learning objects and complying with the IMS CC specification. The repository also adheres to the IMS DRI specification to communicate with other systems.

The EE system selected was Mooshak [13] Mooshak is an open source system for managing programming contests on the Web including automatic judging of submitted programs. The current version (1.6a2) supports the Evaluate service (E-F) [15].

On the IDE side we selected Eclipse. Eclipse is a free and open source multi-language software development environment comprising an IDE and an extensible plug-in system. We tested also the Visual Studio Express IDE on this network with success for C# assignments. In this case we need to install Mono to run .NET applications on the Mooshak server. Mono is a free and open source project to create a standard compliant .NET-

[1] http://code.google.com/p/basiclti4moodle/

compatible set of tools including a Common Language Runtime, C# compiler and others.

4. CASE STUDY

In order to validate Petcha as automatic TA and coordinator of a network of e-Learning systems, we conducted an experiment at ESEIG - a school of the Polytechnic Institute of Porto. First-year Mechanical Engineering students of the course Algorithmics and Programming participated in this experiment. The course aims to widen the students' programming skills using the C# programming language. The course has an average enrolment of 40 students per year divided in two classes. The experiment methodology was the following: only one class (A) used the system while the other class (B) kept the traditional learning approach. The course is organized in two lectures of one hour each and one lab session of 4 hours per week. The experiment occurred in 6 lab sessions. In each lab session the classes (class A with 21 students and class B with 19 students) had 3 exercises to solve. After each lab session we surveyed both classes on the number of solved exercises and the feedback impact. Table 1 aggregates the answers given by students.

Table 1. Statistical data of Petcha usage

Questions	A	B
How many exercises were started in class?	89%	81%
How many exercises were finished in class?	83%	74%
How many exercises solved in class?	82%	66%
How many exercises got feedback?	59%	62%
How many exercises feedback was helpful?	55%	62%

The data collected in the surveys of class A was checked against the logs of Petcha and other systems in the network. An average discrepancy of 4.6% between these two sets of values was found.

The first three lines of Table 1 show that Petcha increases the number of exercises solved by the students. They start and complete more exercises and they have a significantly higher number of exercises effectively solved. In class B the exercises were manually assessed by the human TA. This clearly shows that students solve more exercises when helped by Petcha than when helped by a human TA.

The last two lines of Table 1 show that the automatic feedback provided by Petcha is inferior to the feedback provided by a human TA. Not only the students receive less feedback from Petcha but also this feedback is less helpful than the feedback provided by a human TA. Nevertheless on can argue that the automatic feedback provided by Petcha would be a remedy in a situation where the human TA is not available.

5. CONCLUSIONS AND FUTURE WORK

This paper presents Petcha, an automatic teaching assistant for programming exercises. This tool was conceived to mediate between the teacher and the students and act as an integrator of the best-of-bread systems involved in the process of automatic evaluation of programming exercises. Petcha is a scaffolding tool in the sense that it works with traditional IDEs, helping student to start using the tools they need to program effectively, and can be easily removed when it is no longer helpful. It helps also the teacher in authoring programming exercises for automatic evaluation, including the feedback to provide to students on common error patterns. To achieve these goals Petcha coordinates a network of heterogeneous e-Learning tools, namely program evaluators, learning management environments and learning object repositories.

The main contribution of the research described in this paper is the concept, design and implementation of a tool acting as a teaching assistant for computer programming classes. This tool was designed to coordinate an ensemble of e-Learning systems and the service oriented architecture of the resulting network is also a relevant contribution of this research.

Petcha is currently being used in the practical classes of an undergraduate programming course. The experience gained using Petcha in this context and the experiments designed to assess the impact of this tool were also presented in this paper. These experiments showed an increase in the number of exercises that the students attempted and successfully solve when Petcha replaced a human TA, which was the primary objective of this project. However, these results show also that the automatic feedback provided by Petcha is less effective than that of a human TA. There is clearly room for improving automatic feedback in Petcha, although it can be argued that automated feedback is still a remedy for situations where a human TA is not available.

The current and future work in this line of research contemplates both Petcha itself and the network it manages. Teachers and students reported a set of minor issues on the user interface that are being solved for the next version. There is also a long wish list with features such as: support for Sharable Content Object Reference Model (SCORM) object, support for MathJax for displaying math expressions, improved visualization of evaluation reports, and statistical data on student activity, among others.

Many of the requested improvements are not on Petcha itself but rather on the network of systems and tools it coordinates. To meet these requests we must either provide new features in those system and tools or to integrate new ones in the network. An intended addition is a sequencing and adaptation tool to guide the student through a collection of expository and evaluation resources. Petcha will report the exercise assessment to this new tool that will use it to propose the appropriate content or exercise to the student. Features that could be improved or added to existing systems include: a feedback mechanism using static analysis; a plagiarism detection component; the evaluation of languages that are not strictly programming languages, such as query languages (e.g. SQL), modeling languages (e.g. UML) and user interfaces (e.g. HTML).

6. REFERENCES

[1] Ala-Mutka, K. A survey of automated assessment approaches for programming assignments. *Computer Science Education*, 15(2), pp. 83±102, 2005.

[2] Auffarth, B. and Maite, L. System for Automated Assistance in Correction of Programming Exercises. *In V International Congress University Teaching and Innovation*, pages pp. 104 (1-9)., Lleida (Spain), 2008.

[3] Blumenstein, M., Green, S., Nguyen, A. and Muthukkumarasamy, V. An experimental analysis of GAME: a generic automated marking environment. *In Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, pp 67-71, 2004.

[4] Cheang, B., Kurnia, A., Lim, A. and Oon, W. On automated grading of programming assignments in an academic institution. *In Comput. Educ.*, vol. 41, pp. 121–131, September 2003.

[5] Douce, C., Livingstone, D. and Orwell J. Automatic test-based assessment of programming: a review. *Journal of Educational Resources in Computing (JERIC)*, 5(3), 2005.

[6] Edwards, S. H. and Pugh, W. Toward a common automated grading platform. *In SIGCSE '06: technical symposium on Computer science education*, ACM, 2006.

[7] Engels, S., Lakshmanan, V. and Craig, M. Plagiarism detection using feature based neural networks. *In SIGCSE*, pp. 34–38, 2007.

[8] Free Problem Set (FPS), Official Web site: http://code.google.com/p/freeproblemset/, 2010.

[9] Higgins, C. A., Gray, G., Symeonidis, P., Tsintsifas, A. Automated assessment and experiences of teaching programming. *Journal on Educational Resources in Computing (JERIC)*, 5(3), 2005.

[10] Jackson, D. and Usher, M. Grading student programming using ASSYST. *In Proceedings of 28th ACM SIGCSE Tech. Symposium on Computer Science Education*, San Jose, California, USA, pp 335-339, 1997.

[11] Jena, S. Authoring and Sharing of Programming Exercises. *MsC Thesis*http://scholarworks.sjsu.edu/etd_projects/19, 2008.

[12] Juedes, D. W. Experiences in Web-Based Grading. *33rd ASEE/IEEE Frontiers in Education Conference*, November 5–8, 2003, Boulder, CO, 2003.

[13] Leal, J.P. and Silva, F. Mooshak: a Web-based multi-site programming contest system. *In Software—Practice & Experience*, Volume 33, Issue 6, Pages: 567 - 581, 2003.

[14] Leal, J.P. and Queirós, R. CrimsonHex: a Service Oriented Repository of Specialised Learning Objects. *In ICEIS'09: 11th International Conference on Enterprise Information Systems*, pages 102-113, Italy, May 2009, ISBN: 978-3-642-01346-1.

[15] Leal, José Paulo, Queirós, R. and Ferreira, D. Specifying a programming exercises evaluation service on the e-Framework. *In Advances in Web-Based Learning - ICWL 2010 - 9th Internation Conference*, Shanghai, China, December, 2010, LNCS 6483, pp. 141-150, ISBN 978-3-642-17406-3

[16] Luck, M. and Joy, M. A secure on-line submission system. *In Software - Practice and Experience*, 29(8), pp721-740, 1999.

[17] Malmi, L., Karavirta, V., Korhonen, A., Nikander, J. Experiences on automatically assessed algorithm simulation exercises with different resubmission policies. In Journal on Educational Resources in Computing (JERIC), 5(3), 2005.

[18] Mandal, A.K., Mandal, C. and Reade, C.M.P. Architecture Of An Automatic Program Evaluation System. *In CSIE Proceedings*, 2006.

[19] Mandal, C., Sinha, V.L. and Reade, C. M. P. A Web-Based Course Management Tool and Web Services. In Electronic Journal of E-Learning, Vol 2(1) paper no. 19, 2004.

[20] Mansouri, F.Z., Gibbon, C.A., Higgins, C.A. PRAM: prolog automatic marker. *In Proceedings of ITiCSE'1998.* pp.166~170, 1998.

[21] Pisan, Y., Richards, D., Sloane, A., Koncek, H. and Mitchell, S. Submit! A Web-Based System for Automatic Program Critiquing. *In Proceedings of the Fifth Australasian Computing Education Conference (ACE 2003)*, Adelaide, Australia, Australian Computer Society, pp. 59-68, 2003.

[22] Queirós, R. and Leal, J.P. PExIL: Programming Exercises Interoperability Language. *XATA 2011 – XML, Aplicações e Tecnologias Aplicadas*, Junho 2011.

[23] Queirós, R. and Leal, J.P. A Survey on eLearning Content Standardization. *4th WSKS*, 2011, Mykonos, Greece.

[24] Reek, K. A. The TRY system or how to avoid testing student programs. *In Proceedings of SIGCSE*, pp 112-116, 1989.

[25] Rehak, D. R., Mason, R. Keeping the learning in learning objects. *In Littlejohn, A. (Ed.) Reusing online resources: a sustainable approach to e-Learning*, 2003. (pp.22-30).

[26] Saikkonen, R., Malmi, L. and Korhonen, A. Fully automatic assessment of Programming exercises. *In Proceedings of the 6th Annual Conference o Innovation and Technology in Computer Science Education (ITiCSE'01)*, Canterbury, United Kingdom, pp. 133-136, 2001.

[27] Striewe, M. and Goedicke, M. Visualizing Data Structures in an E-Learning System. *In Proceedings of the 2nd International Conference on Computer Supported Education (CSEDU)*, Valencia, Spain, volume 1, pages 172-179, 2010.

[28] Tang, C. M., Yu, Y. T., & Poon, C. K. Automated systems for testing student programs: Practical issues and requirements. *In Proceedings of the International Workshop on Strategies for Practical Integration of Emerging and Contemporary Technologies in Assessment and Learning (SPECIAL 2009)*, pp. 132±136, 2009a.

[29] Tang, C. M., Yu, Y. T., & Poon, C. K. An approach towards automatic testing of student programs using token patterns. *In Proceedings of the 17th International Conference on Computers in Education (ICCE 2009)*, pp. 188±190, 2009b.

[30] Tang, C.M., Yu, Y.T. and Poon, C.K. A Review of the Strategies for Output Correctness Determination in Automated Assessment of Student Programs. *In Proceedings of Global Chinese Conference on Computers in Education,* 2010.

[31] Trætteberg, H, Aalberg, T. . JExercise: A specification-based and test-driven exercise support plug-in for Eclipse. *In Proceedings of the 2006 OOPSLA Workshop on Eclipse Technology eXchange, ETX 2006* (2006), 70-74.

[32] Verdú, E., Regueras, L.M., Verdú, M.J., Leal, J.P., Castro, J.P. and Queirós, R. A Distributed System for Learning Programming On-line. *In Computers & Education Journal*, 2011, ISSN 0360-1315

[33] Verhoeff, T. Programming Task Packages: Peach Exchange Format. In Olympiads in Informatics, 2008. Vol. 2 192-207.