Jutge.org

Omer Giménez* Jordi Petit* Salvador Roura*

* Google, inc.
* Universitat Politècnica de Catalunya

September 5, 2011

Abstract

Jutge.org is an open access educational online programming judge where students can try to solve more than 800 problems using 22 programming languages. The verdict of their solutions is computed using exhaustive test sets run under time, memory and security restrictions. By contrast to many popular online judges, Jutge.org is designed for students and instructors: On one hand, the problem repository is mainly aimed to beginners, with a clear organization and gradding. On the other hand, the system is designed as a virtual learning environment where instructors can administer their own courses, manage their roster of students and tutors, add problems, attach documents, create lists of problems, assignments, contests and exams. This paper presents Jutge.org and offers some case studies of courses using it.

1 Introduction

It is a well established fact that practice is fundamental to learn computer programming. Whether at secondary, high school or university level, instructors assign programming problems to beginner students in order to help them acquire this skill.

These problems often require devising an algorithm and writing a program that, given some inputs, produces the corresponding outputs. Most of these problems put their emphasis on understanding rather than on technical knowledge: the problems are not solved through some "magic" command, but, rather, on thinking how to algorithmically combine a few basic instructions and data structures that appear in just about any modern programming language.

For instructors, correcting programming assignments is tedious and error-prone. However, due to their nature, programming problems are ideal candidates for automated assessment. A review on automatic assessment of programing problems can be found in [9] and recent experiences include [8, 12, 14].

That said, the current running horses of automatically correction systems are the *online judges* in the Internet. Online judges are web based systems that offer a

repository of programming problems and a way to submit solutions to these problems in order to obtain a verdict on their behaviour when run on different public and private data sets. Some fine and popular online judges include UVa [6], Timus [4], Sphere [3], CodeChef [1], TopCoder [5], ...

However, the main purpose of the aforesaid systems is to help their users to train themselves for several types of contests, such as the ACM ICPC or, even, to help companies recruiting programmers (TopCoder). As a consequence, these judges are competitively oriented, and the difficulty of most of their problems ranges from advanced to very difficult. Even though experiences with competitive learning have been reported [13, 15], overall these sites are not adequate for beginners and do not provide much help for their instructors.

In order to remedy this situation, we present Jutge.org, our online judge built with an educative aim. Jutge.org innovates in a number of aspects. First, and in contrast to other online judges, Jutge.org is designed both for students and instructors. Integrating in a coherent way ideas from both traditional online judges and from learning management systems as Moodle [2], instructors can manage their own courses, add their own problems, attach documents, create assignment lists, monitor the progress of the students, update rosters of students and tutors and interact with them. Second, the repository of problems at Jutge.org is clearly organized by topics and graded by difficulty. The problem collection enables a systematic progress, largely independent of the programming language of choice. Thus, instructors can tailor their programming courses using a large base of problems according to their needs and preferences.

The paper is organized as follows: First, we sketch the interface of Jutge.org, showing its main resources and presenting two scenarios. Then we survey its architecture, stressing security concerns. Next, we present its repository of graded problems. Finally, we describe our experience using the system in several settings, draw some conclusions, and advance ideas towards future work.

Jutge.org is at https://www.jutge.org and has an open access policy. All features can be tried without registering using the demo account. The spelling of Jutge.org follows Catalan, not English.

2 Interface

In order to present the possibilities of Jutge.org, we first introduce the main types of users and the main teaching resources available at Jutge.org; then, we present two typical scenarios of use.

2.1 Types of users and teaching resources

Users in Jutge.org can have different roles: *Students*, *Tutors*, *Instructors* and *Administrators*. These roles are not mutually exclusive. Jutge.org is free and will remain

free for all students and tutors. Currently, Jutge.org is also free for instructors.

The main resource of Jutge.org are *problems*. Problems are identified by a unique short code and consist of a problem statement that describes the task to perform, some public test cases (sample input and sample output) and some private test cases. For instance, problem P29212 is titled "Modular exponentiation" and can be found at https://www.jutge.org/problems/P29212.

Most of the problems can be solved in any programming language that has been set up in the judge. Currently, 22 programming languages with 31 different compilers are available. These include[1] C++, C, Java, Python, Scheme, Pascal, ...

Almost all problem statements in Jutge.org are offered in English. As Jutge.org supports multilingual problems, most problems are also offered in Catalan or in Spanish (or both). Statements are available in HTML, PDF and PS.

The teaching resources of Jutge.org are primarily organized in *courses*. At their turn, courses can contain *lists of problems*. For instance "Algorithms" is a public course that contains a list called "Divide and conquer" that, in turn, contains several problems including "Modular exponentiation". A problem can be in many lists and a list can be in many courses. For instance, "Modular exponentiation" is also in the "Problems on Mathematics" course. Moreover, courses can contain *documents* in HTML format.

In addition, instructors can also add problems, and organize *exams* and *contests* within courses. Scripting is also possible by the way of *webservices*. Because of space reasons, these features will not be much discussed in this paper; see [7] for technical details.

2.2 First scenario: Standalone user

Let us consider the following scenario: Some user —let's call her Lisa— wishes to use Jutge.org to learn to program or, maybe, to improve her programming skills.

In order to use Jutge.org, Lisa first creates her account. Once logged in, Lisa will find that the system already offers her some public courses and some public lists of problems (see Section 4.1). As any other user, she can enroll any of those courses.

After browsing the contents of the public courses, Lisa can try to solve a problem. To do so, Lisa reads the problem statement and considers its sample test cases. At this point, Lisa designs and develops a solution for the problem in her own computer, compiling it, and validating it using the sample test cases and other test cases that she can think about herself. Finally, Lisa uploads the source code file to the online judge, which returns its verdict, typically in less than five seconds.

In order to emit a verdict, the engine of the online judge tests the solution

[1] See https://www.jutge.org/documentation/compilers for the complete list of compilers and .../verdicts for the complete list of verdicts.

submitted by Lisa using a set of private test cases. Possible verdicts include "Accepted", "Wrong answer", "Execution error", and others[1].

Lisa can submit to the judge as many solutions as desired. There is just a reasonable limitation in the number of allowed submissions in order to dissuade users from using the judge as an online compiler. All the submitted source code and associated verdicts together with satellite data as submission date are stored in the judge so that Lisa can retrieve them later.

In addition to the capacities related to submitting and retrieving solutions (which are more or less similar to other online judges available), Lisa can update her profile, enroll and un-enroll courses, browse the courses, lists, documents, exams and problems, and even link her Jutge.org account with Twiter and/or Facebook so that the judge will automatically post her accomplishments in her social network.

2.3 Second scenario: An universitary course

We consider now another typical scenario where Pr. Oak is responsible for an introductory course to programming at her university for the next semester. 400 students (including Carla) will take her course, and a dozen of teachers (including Bob) will also assist her during the laboratory sessions. Pr. Oak is convinced that recursion is very important, so her course will make emphasis on this point.

In order to set up her course using Jutge.org, Pr. Oak will get an account with instructor privileges and proceed as follows:

First, Pr. Oak will create a new course, "CS1 2011" say. Then, she will create some lists of problems and will populate them with the public problems already offered by Jutge.org according to her preferences. The system already offers so many problems that she does not need to create new problems on her own. Her course may contain, for instance, some lists called "Introduction", "Procedures", "Recursion" and "Backtracking" in accordance to the course syllabus. As she has already prepared some notes she plans to distribute, she will place them in new documents, or will link them from these documents. At this point, she is ready to invite all her students by providing their official email address at the university. All invited students will receive an email stating that they can enroll the "CS1 2011" course. Naturally, only the invited users are allowed to join her course. In the same way, Pr. Oak will also invite her assistant teachers, who will be designated as tutors for this course.

All through the semester, Pr. Oak can freely modify the information contained in her course. These updates will be seen in real time by all her students and tutors.

At some point, it may happen that Carla has some doubt about a particular problem that she is not able to solve. So Carla contacts Bob my email to state her difficulties and ask for help. In order to help students, tutors can use a special feature of Jutge.org called *supervision*: Supervision enables Bob to impersonate Carla's account and access her problems and submissions related to the "CS1 2011" course.

By looking to her submissions and inspecting her code, Bob can spot her mistakes, answer her doubts and offer further guidance.

Note that, because of privacy issues, a tutor can only supervise students that have enrolled his courses and, moreover, supervision is limited to the problems that this course contains. Moreover, during supervision, the supervisor tutor cannot perform any action on behalf of the supervised student.

We pursue our scenario by imagining that, at some point, Pr. Oak wants to gather data on the overall progress of her students. To help her, Jutge.org offers a wealth of diagrams and statistics such as the ones shown in Figure 2.3.

Additionally, Pr. Oak is able to create exams or contests for the students in her course and Jutge.org also offers her other minor but practical functionalities that alleviate her administrative burden. For instance, the solutions that the students submit during the exams are easily distributed to the tutors, so that they can grade those solutions according to their own method.

We finish our scenario by noting that, on the next year, Pr. Oak could again be responsible for the same course but, of course, with different students. So, she will only need to copy "CS1 2011" to a new course "CS1 2012", invite a new roster of students and, maybe, new tutors. After this copy, the lists and documents are preserved but, naturally, can be further modified.

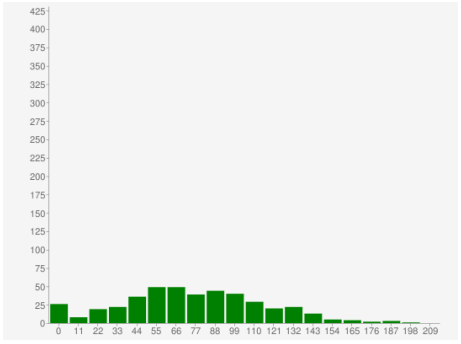
3 Architecture and security

In this section we survey the architecture on which the Jutge.org system is built, which is largely dependant on the possible attacks such a system must deal with, the requirement to offer verdicts in almost real time, and the need to make such a system scalable.

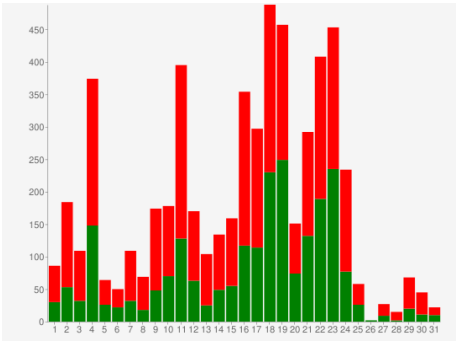
3.1 Security concerns

As any IT system, an online judge must cope with several threats. However, unlike many other systems, an online judge will welcome all users to submit their potentially malicious codes and execute them on their behalf. A nice analysis on the risks of such environments can be found in [10]; these include: Forcing a high compilation time, consuming resources at compilation time, accessing restricted information, misusing the network, modifying or harming the testing environment, circumventing the time measurement, exploiting covert channels, misusing additional services, and exploiting bugs in the OS.

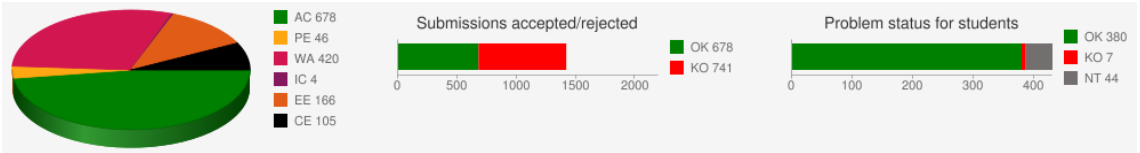
Consequently, our system must provide a secure execution environment to execute arbitrary programs written in a myriad of programming languages without comprising the stability of the system nor its confidential information.



(a) Histogram of solved problems by students. Hor. axis: number of problems; vert. axis: number of students.



(b) Number of submissions for each day of May 2011. Green: accepted submissions; red: rejected submissions.



(c) Results for problem P67723 (Euclides algorithm): At left, distribution of verdicts; at center, proportion of accepted/rejected submissions; at right, number of students with the problem accepted (OK), rejected (KO) and not tried (NT).

The actual data in the three subfigures corresponds to the “Programació 1” course during spring 2011 at FIB.

Figure 1: Some diagrams that Jutge.org offers to instructors.

3.2 Efficiency and scalability

Users expect online judges to emit their verdicts in almost real time. In our case, we aim to correct almost all submissions in less than ten seconds. Such a rate can be considered standard.

In order to guarantee these correction rates as the number of users increases, we have designed a distributed system that is scalable by adding more servers to it. Here, a key problem is that our resources only allow adding servers from time to time and, because of this, these servers will have different speeds and capacities: we cannot consider that all of them are equal and, so, CPU time limits cannot be fixed in advance. In any case, users should not be aware of these differences.

3.3 Infrastructure and network architecture

The current Jutge.org infrastructure contains six Linux servers. One of these servers (the master or gateway) is the one that is used as the front-end: it offers the web, stores the database and maintains the submissions queue. The other servers are the back-end and are used as slave machines to process submissions. A separate backup and monitoring system is kept.

The overall distributed architecture is shown in Figure 2.

The master is directly connected to the Internet. For security reasons, the slaves are only connected to the master using a private network. In this way, slaves do not have a direct Internet access.

Moreover, the slaves do not process the submissions on their own, but rather rely on different virtual machines that are periodically restarted from scratch using inmutable disks. Each slave machine runs as many virtual machines as cores it has. Also, these slaves are used as a firewall to make more difficult escaping from the virtual machines they provide.

3.4 Correction of a submission

Users submit their submissions from the web server. These are sent to the queue manager, which will dispatch them to the virtual machines and return their correction.

According to some parameters that influence the priority of the submissions (e.g., exam submissions must be corrected before regular ones) and the load of the slaves machines, the queue manager sends the submissions to the virtual machines in the slave machines as soon as possible. In fact, the queue manager not only sends the actual submission, but also some related data as the problem test cases and the reference solution. In any case, each virtual machine just corrects one submission at a time.

Once in a virtual machine, submissions are corrected according to a *driver* that depends on the problem type. Roughly, the standard driver computes the verdict

of the submission in two phases:

1. In the first phase, the reference solution is checked for correctness. That is, it is compiled and executed on all the test cases. In the event the reference solution did not work, an "*Internal error*" verdict would be issued. During the execution of the reference solution, CPU timings, wall-clock timings and memory measurements are performed.

2. Then, in the second phase, the submitted solution is checked for correctness as follows: First, the submission is compiled; in case of an error, the verdict is "*Compilation Error*", and the complete error will be reported to the user. If compilation succeeds, the compiled program is executed, as many times as test cases the problem has. If any of these executions aborts, the verdict is "*Execution Error*", together with a short explanation of the cause of the error (segmentation fault, CPU time exceeded, etc.). If not, then all the produced outputs are checked against the correct outputs (Section 4.3 gives more details in the possible ways to do it). If the checking is as expected, then the verdict is "*Accepted: Test cases passed*". If some of the files differ, then an additional check is performed to guess whether the differences are only minor (spaces or carriage returns, lower or upper case differences, etc.), in which case the verdict is "*Presentation error*". If this relaxation does not account for the output difference, then the verdict is "*Wrong answer*".

3.5 Secure environment

In order to protect the integrity of the system, all the above mentioned executions of programs (including compilation steps) are performed in a sandbox environment that includes switching to a low-privilege nobody user, limiting file system access by a chroot jail, monitoring system calls and using setrlimits (similar in spirit to [16]), and restricting CPU-time, clock-time and memory usage.

The queue manager waits the correction of the virtual machine. If it takes too long, an error is reported to the administrator (failure or possible attack) and the machine is rebooted. In case of success, the verdict and its related information is stored in the database.

Ideally, once a virtual machine has corrected a submission, this virtual machine would be rebooted. However, rebooting after each correction takes too much time and our experience restoring machines from a frozen state is negative. So we resort to just killing all nobody processes and only rebooting machines after a dozen of corrections so as to amortize the reboot time.

All connections use SSH or HTTPS. The only step that requires root privileges is the switch to the chrooted and unprivileged user environment, which is done with a setuid program.

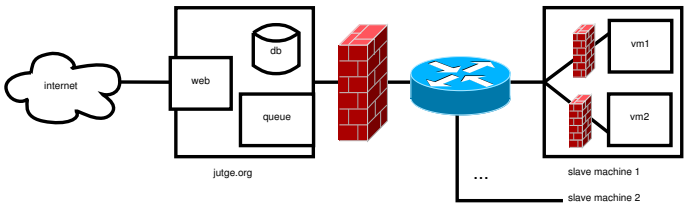


Figure 2: Network architecture and security

4 The problems

Most of the problems publicly available in Jutge.org have been written by the authors in order to teach their respective courses at Universitat Politècnica de Catalunya (UPC). Many of these problems are at introductory and medium levels, while the average level of the problems in other online judges as UVA and SPOJ is much higher. This reflects that Jutge.org is an educative online judge.

In this section, we first present the problem collections publicly available. We then survey which kinds of problems exist.

4.1 Problem collection

Right now, Jutge.org contains more than 800 problems publicly available. These problems are organized in the following courses:

Learning to program: This course is aimed to students who want to learn programming from scratch using an imperative language. This course is further organized in different graded lists: Introduction, First loops, Loops inside loops, First procedures, Recursivity, Sequences, Consolidation I, Unidimensional arrays, Multidimensional arrays, Tuples, Fundamental algorithms, and Consolidation II.

Algorithms: This course is a continuation of the previous one, placing its emphasis on the use of data structures and the application of basic algorithmic techniques. It currently contains the following lists: Backtracking (Subsets), Backtracking (Permutations), Backtracking (Consolidation), Dynamic programming (1), Binary search, Divide and conquer, Sets and maps, Stacks, Queues and priority queues, Greedy algorithms, Graph algorithms, Parsing problems, More backtracking, Brute force, Dynamic programming (2), and Other problems.

Problems from the OIE contests: This course gathers all problems used in the Spanish Olympiads (see Section 5). Thus, the problems in this course are oriented towards users with some programming experience and their difficulty ranges from easy to hard.

Problems from the UPC contests: This course gathers all problems used in the programming contests organized by the Universitat Politècnica de Catalunya (UPC) to train and select students to participate in the ACM ICPC contests. Thus, the

problems in this course require some knowledge on algorithms and sound programming. This course contains most of the more difficult problems in this judge.

Problems from the FME Contests: This course gathers all problems used in the programming contests organized at the Facultat de Matemàtiques i Estadística (FME) among the students of a second semestral course on algorithms. The problems in this course also require some knowledge on algorithms and programming but are, in general, easier than those of the UPC contests.

Problems on Mathematics and **Problems on Physics:** These courses include problems related to mathematics and physics. They are still under development. Their goal is to provide interest on programming to people in these areas.

Many of the problems have been tested extensively by UPC students during the latest five years. As a consequence, we are quite confident about their statements, solutions and test cases.

4.2 IO problems versus procedural problems

Many online judges follow the UVa model and ask for programs that read some input and write some output depending on that input. Other judges, as TopCoder, ask for some class or function with a fixed interface on their arguments.

Jutge.org offers both approaches. Most of the problems follow the IO model, but other problems explicitly ask for a conformant function or class. Indeed, procedures, functions and classes are a fundamental concept taught in introductory courses.

For instance, problem P22654 asks to write a C++ procedure `void decompose(int n, int& h, int& m, int& s)`; that, given a quantity of seconds `n`, computes how many hours `h`, minutes `m` and seconds `s` it represents. Likewise, problem P48763 asks to implement some methods for a *Clock* class.

Problems that ask to write a particular function also may arise when the problem setter wants to check that an efficient sublinear solution has been used. For instance, dichotomic search (P81966).

In any case, in order to test candidate solutions to these problems, the problem setter must write a client program that will call the function written by the student. A difficulty we face with this type of problems is that both their statement and their associated client programs must take into account the target programming language.

4.3 Checkers

Most problem statements on online judges are written in an unambiguous way, specifying the set of valid inputs and their corresponding unique correct outputs. The validation process of a submission is easy: the outputs of each test case produced by a candidate solutions must exactly match the outputs produced by the

reference solution. As said in Section 3.4, when the differences are only minor a "*Presentation error*" verdict is reported.

In Jutge.org, most of the problems are also of this type, but the system also offers some variations (internally implemented by *checkers*) that we survey below.

Lousy checkers. In many cases, enforcing a strict format of the output is unnecessary. The problem setter may not care whether words must be separated by one space, two spaces or a blank line. Likewise, the case of the letters may not matter. Problems whose output should take into account these considerations use the *lousy checker*. So, the effect of this checker is turning "*Presentation error*" verdicts into "*Accepted*" ones.

Elastic checkers. In many enumerative problems, the order of the output is also irrelevant, provided it contains all what it must contain but not more. For instance, problem P79494 asks for a list of all possible completions of a sudoku puzzle. Here, the order in which all solutions are written is not essential and, in fact, forcing a particular order would complicate the statement and negatively affect the creativity of the student. Analogously, in problem P18957, where all subsets of $\{1, ..., n\}$ must be printed, the order in which these 2^n subsets are written is not important and, furthermore, the order in which the elements of each subset are written is also irrelevant.

To handle this type of problems, Jutge.org offers *elastic checkers* that, given a specification of the separators of the elements, take these considerations into account. We note that these checkers can also report "*Presentation errors*" by default.

Epsilon checker. In some types of mathematical problems, floating point precision can be an issue. In some cases, these issues render some problems too unstable to be written (for instance, *are three given points aligned?*). In other cases, when the output of a problem is a floating point number, an approximation can be tolerated. To address this issue, the *epsilon checker*, given an ε value, tolerates and absolute error of ε between the values in the reference output and the candidate output.

For instance, problem P37619 asks to compute the sine of an angle using Taylor series.

External checkers. In some problems, solutions are not unique. For instance, the Minimum Spanning Tree problem can have many optimal solutions for each weighted graph given as input. Jutge.org offers a powerful way to check the output for this type of problems using *external checkers*. An external checker is another program (written by the problem setter) that, given an input of the problem and given the output of a candidate solution on that input, decides whether this solution is correct or not. For instance, in the case of the MST, the checker would first read the weighted graph, then compute the weight of a MST using its own solution; afterwards, it would read the edges listed by the output of the candidate solution and, finally, would check that these exists, that they really form a spanning tree and, that the sum of their weights equals the one found by the reference solution.

P43694 is such a problem.

5 Experience

In this section we describe several case studies using the system.

5.1 Programació-1

Jutge.org is an evolution of our previous experience with the "Jutge de P1", an on-line judge specially aimed for and restricted to the students of the course Programació-1 taught at Facultat d'Informàtica de Barcelona (FIB). P1 is a first-year, compulsory, one semester programming course that assumes no previous knowledge on computers in general nor programming in particular. In average, 400 students take this course each semester.

A detailed account on the experience organizing P1 using the judge during six semesters is given in [11]. The main characteristic of this course is its very strong emphasis placed in solving programming problems. Indeed, the fundamental objective of P1 was stated as: *"a student passing P1 must be able to confidently code correct, readable programs solving problems of elementary difficulty"*. All P1 exams were online.

Consequently, this course required, on the one hand, an extensive problem collection, carefully selected and ordered, and on the other hand, a fully automatic judge to validate in real time the programs submitted by the students. Our opinion is that using an on-line judge offers several advantages for such a course:

- As a learning tool, it is a useful device to learn programming, since it contains an extensive list of exercises, automatically evaluates the solutions submitted by students throughout the course and gives an immediate verdict about their correctness (relying in very exhaustive test cases).

- As a tool for helping on the evaluation process, the on-line judge makes it possible to grade students precisely in the competence that they must acquire, that is, solving simple programming problems. This includes reading and understanding the statement of a problem, thinking about an algorithm solving it, coding it, and checking that it works. In addition, the judge is objective and unbiased.

- As a tool for monitoring students progress, the on-line judge provides a source of objective, untainted data from a hitherto unknown depth to us. The instructors can easily access this data, both as global statistics on all students enrolled, or as particular data restricted to a particular student or programming problem.

With respect to the observed results, [11] shows that there is a clear correlation between the amount of problems the students solve and the grades they obtain; also, that a great majority of the students that solve the laboratory and homework problems pass the course. Unfortunately, the data collected on the on-line judge

made clear that the effort that most students dedicated to the course was far from the workload required according to the number of ECTS credits. As a sad but logical result, a high number of those students failed P1.

5.2 Courses on algorithms

Similarly, Jutge.org has also been used for students in their second year/course of studies, emphasizing a syllabus more related to data structures and algorithms. In particular, Anàlisi i Disseny d'Algorismes and Estructures de Dades i Algorismes at FIB and Informàtica-2 and Algorísmia at FME. In these courses, about a third of the grade is based on an online exam using the judge.

5.3 Various competitive experiences

While Jutge.org emphasises education, it can also be used as an online system to organize programming contests.

The most visible event hosted by Jutge.org is the organization since 2007 of the Olimpiada Informática Española (OIE), which is an annual programming contest open to students in Spanish high schools. The OIE is organized in two rounds: In the first round, contestants participate in an online contest organized via the Internet. In the second round, the top 25 students are invited to participate, in-place, in a final contest. The four OIE champions represent Spain at the International Olympiad in Informatics. Participants in the OIE also use Jutge.org in order to train themselves, access the problems of the past years and practice in warm-up contests. For historical reasons, OIE participants use another web front-end (olimpiada-informatica.org) to access our online judge.

Our system has also been used since 2003 to train and select the students from our university (UPC) that participate in the regional ACM ICPC contests. Analogously, the Universitat Pompeu Fabra has recently used Jutge.org in order to organize their contests.

5.4 Promotion of CS in 2ary and high school

The authors of Jutge.org are also involved in courses that promote computer science and computational thinking at secondary and high school levels. Recently, three such courses have taken place, sponsored by the Joves i Ciència program of CatalunyaCaixa, whose mission is to enhance the scientific activity of young students with interest, motivation and talent for science. These are intensive courses targeted to high school students with top performance but with no (or minimal) experience in programming. The courses take place for one week just at the beginning or the end of the school summer holidays and are taught by university professors with the help of university students.

The syllabus of these courses is similar to the one that we have described for P1, probably with less width and depth, but combining a strongly practical approach with a minimal theory to guarantee scientific rigour. Their development is purposely set under an entertaining mood.

The feedback received by the Joves i Ciència program after these courses is that the students are very satisfied with the experience and specially appreciate its practical approach. Not surprisingly, some of them find the course too intense, while others enjoy such an intensive course.

6 Conclusion

In this paper we have surveyed the resources, the functionality and the implementation of Jutge.org, an online judge specially built with an educative aim. We have also related it to various teaching experiences.

Built on a base of more the 800 clearly organized and graded problems, the main features of Jutge.org for instructors are the automation of the correction process, the creation and maintenance of their own courses and rosters, the management of their assignment lists, and the access to instant information about the progress of their students. The main features of Jutge.org for students are the possibility to enroll in their instructor courses and in public courses, to get instant feedback on their solutions, and to practice with many different programming languages.

The different experiences we have reported include the instruction of two massive courses (a first programming course and a course on basic algorithms and data structures) that have involved thousands of university students, the instruction of intensive courses to promote computer science among high school students, and the organization of several events, such as internal university contests and the Spanish Olympiads in informatics. These experiences, which already account for half a million of submissions, confirm the flexibility, robustness and reliability of Jutge.org.

For the near future, further system deployment is envisaged. On one hand, we are considering the use of cloud resources to distribute our correction load. On the other hand, we are working towards data mining the most discriminative and small data sets for each problem, based on the many wrong submissions we have historically received. We expect that distilling and offering these highly discriminative inputs to tutors would boost their productivity at helping students.

References

[1] CodeChef. http://www.codechef.com.

[2] Moodle. http://moodle.org.

[3] Sphere Online Judge. http://www.spoj.pl.

[4] Timus Online Judge. http://acm.timus.ru.

[5] TopCoder. http://www.topcoder.com.

[6] UVa Online Judge. http://http://uva.onlinejudge.org.

[7] A. Caterineu and J. Petit. Jutge.org: Instructor's Guide. http://www.jutge.org/documentation/guide.pdf, 2011.

[8] B. Cheang, A. Kurnia, A. Lim, and W.-C. Oon. On automated grading of programming assignments in an academic institution. *Computers & Education*, 41(2):121–131, 2003.

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

[10] M. Forišek. Security of Programming Contest Systems. In V. Dagiene and R. Mittermeir, editors, *Information Technologies at School*, pages 553–563, 2006.

[11] O. Giménez, J. Petit, and S. Roura. Programació 1: A pure problem-oriented approach for a CS1 course. In C. Hermann, T. Lauer, T. Ottmann, and M. Welte, editors, *Proc. of the Informatics Education Europe IV (IEE-2009)*, pages 185–192, Freiburg, 2009.

[12] M. Joy, N. Griffiths, and R. Boyatt. The BOSS online submission and assessment system. *ACM Journal on Educational Resources in Computing*, 5(3), 2005.

[13] A. Kosowski, M. Malafiejski, and T. Noinski. Application of an online judge & contester system in academic tuition. In *ICWL'07*, pages 343–354, 2007.

[14] A. Kurnia, A. Lim, and B. Cheang. Online judge. *Computers & Education*, pages 299–315, 2001.

[15] M. Revilla, S. Manzoor, and R. Liu. Competitive learning in informatics: The UVa online judge experience. *Olympiads in Informatics*, 2:131–148, 2008.

[16] B. Yee et al. Native client: A sandbox for portable, untrusted x86 native code. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2009.