Olympiads in Informatics, 2012, Vol. 6, 31–43
© 2012 *Vilnius University*

Teaching Programming and Algorithm Design with Pythia, a Web-Based Learning Platform

Sébastien COMBÉFIS, Vianney le CLÉMENT de SAINT-MARCQ
Department of Computer Science Engineering, Université catholique de Louvain
Place Sainte Barbe 2, 1348 Louvain-la-Neuve, Belgium
e-mail: {sebastien.combefis, vianney.leclement}@uclouvain.be

Abstract. In Belgium, there are no or very few programming and algorithm design courses at secondary schools (12–18 years old). Students who can program are self-learners. The selection process for the Belgian delegation for the IOI, that has been set up two years ago, takes this situation into account. More initiatives should be taken to introduce computer science in secondary schools. This paper presents Pythia, a solution grader for programming and algorithm design problems based on a web application. Following the *learning by doing* motto, the proposed framework provides an accessible, usable, effective way to learn programming. To compensate for the lack of teachers with programming or algorithm design skills, Pythia delivers direct feedback to the students. The paper describes the design of the courses and the architecture of the tool. As future work, the proposed teaching technique has yet to be tested at large and evaluated.

Key words: solution grader, teaching, programming, algorithm design, learning platform, online course.

1. Introduction

In Belgium, there are no or very few programming and algorithm design courses taught at secondary schools. Nevertheless, some 12–18 years old Belgian pupils are good at programming and designing algorithms, but they do not know it (Combéfis and Leroy, 2011). As far as we know, there are very few opportunities to encourage such people to learn programming. The contestants participating to the Belgian Olympiad in Informatics (be-OI) are mainly pupils with self-taught programming experience.

When taught in secondary schools, computer science is taken in charge by teachers from other disciplines such as mathematics or physics. The same situation happens in many other countries. Teachers do not always have the suited experience to be able to teach computer science. Sometimes they are even asked to teach computer science but they just do not know what to do.

Discussions with the contestants of the be-OI raised several ideas. One of them was to gather on a website existing online programming contests and to allow the contestants to publish their scores. As such, they can compare with each other and develop a competition spirit. Another idea was to propose online programming courses targeted at people with zero background in programming and algorithm design.

The main problem addressed by this paper is to find and develop a solution providing an easy way for people to learn programming. The solution should not require any difficult installation and configuration of the personal computer. Moreover, the solution should be flexible enough so that pupils can learn autonomously at home and so that secondary school teachers could use it to teach in classes.

As an attempt to implement the ideas given above, this paper presents Pythia, a web-based solution grader application. Pythia provides both online programming and algorithm design courses, and a collection of problems users can solve and compare with each other. Code submitted by students is executed safely inside a sandbox. The Pythia tool is actually more general than what it is used for in the context of this work. The tool can be used for many other purposes, such as providing automatic grading for students' homework or supporting online contests for example.

Section 2 draws up a related work presenting some existing solutions to address the problem of providing means to learn programming. Besides the description of courses and problems in Section 3, this paper shows the technical aspects of Pythia in Section 4. Section 5 details the plan for evaluating the tool. Finally the last section concludes the work and draws up some perspectives.

2. Related Work

Several initiatives exist to support people in learning programming. For example programming contests are a good motivation for people to learn and improve their skills. However, the contestants must learn by themselves if no trainings are organised. Contests like olympiads are also targeting excellence, excluding less-skilled people. A positive aspect of contests is the discussions it triggers between contestants, especially for non-online contests.

A computer science festival for young people is another initiative. In Belgium, such festivals sensitise people to computer science without using computers. Useful resources include CSUnplugged (Bell *et al.*, 2009) and Scratch (Maloney *et al.*, 2004). School teachers come to such events with their pupils. They follow some activities and learn at the same occasion. Teachers often continue the activity with the pupils back in their classroom. While festivals spark great interest, they require a lot of specially-trained human resources. Due to their large audience, festivals also do not dive deep into the details. As such, they are adequate for advertising purposes, but not for learning purposes.

A third kind of initiative is online learning. Web-based applications avoid the need for the learner to install anything on his/her computer. There are plenty of online applications like *"Rubymonk"*, *"Try Ruby"* or *"Try Python"*. The main goals of such applications is usually to learn a specific programming language rather than general programming or algorithm design skills. The learner can execute small snippets of code to get a grasp of the language. Some sites, like *"Codecademy"*, try to learn programming as a general skill. However, *"Codecademy"* does not teach algorithm design and lacks support to provide quality feedbacks to learners.

The solution proposed in this paper shares the technology of learning sites. In contrast to other sites, if focuses on learning programming and algorithm design skills. Special care has been taken to provide a language-agnostic framework for the safe execution of students' code.

Systems for executing students' code in a safe way have been proposed in the literature. The hackzor system is used for online competition contest judging. The Moe Contest Environment (Mareš, 2009) is targeted at competitions in the spirit of the International Olympiads in Informatics. Moe sandboxes forbids dangerous system calls. Pythia implements a less restrictive approach by leveraging virtual machines. Such approach is shared by the uevalrun system. However, uevalrun lacks the surrounding framework provided by Pythia.

3. The Problems

The proposed learning platform basically consists of two activities. The learner can follow a course where he/she is guided through several lessons that will teach him/her programming and algorithm design concepts. The goal of the lessons is to bring theory to the learner and to allow him/her to practice directly. The learner can also try to solve isolated problems. The goal of those problems is to train algorithm design by allowing the learner to solve problems. The two kinds of activities are uniformed around the concept of *task*.

This section reviews the two kinds of activities and illustrates them with examples. Some intuition about how to design such problems and about what is possible with the proposed learning platform is also provided.

A *task description* is an XML file which contains the following information:

– Basic information: author, title and difficulty level.
– Constraints: programming language, maximal execution time and memory. Optionally, a limit on the total number of submissions and/or on the number of submissions per day can be specified. A deadline can also be given.
– Task contents: context and questions.

Besides the above information, a task description contains two programs. The *template program* contains placeholders to be filled with the student's answers. The completed program will be run in a sandboxed environment. The result of the execution will be analysed by the second program, i.e., the *analyser script*, whose goal is to generate a feedback for the user.

We now explain how isolated problems (Section 3.1) and courses (Section 3.2) are designed using task descriptions. Section 3.3 expands on the feedback, an important point in the learning process.

3.1. *Isolated Problems*

Isolated problems can be solved by the students independently from any other problems.

The key elements of isolated problems are:

1. The problem should be put in context and explained with a concrete situation.
2. Simple instances should be provided to be solved by hand in order to get the intuition of the problem.
3. Larger instances should be provided to be solved with an algorithm.

The first item is required to increase the motivation of the user to solve the problems. The context links the abstract problem to concrete situations the student encounters in everyday life. The user thus gains more value to the activity of solving the problem. It is essential for the student to perceive an interest and a pedagogical utility with respect to his goals for the task he is doing (Viau, 1998). According to Viau, the student's perception of his controllability of the activity is also important in the motivational dynamics. Such concern is addressed by allowing the student to choose the problems by himself according to his level and tastes.

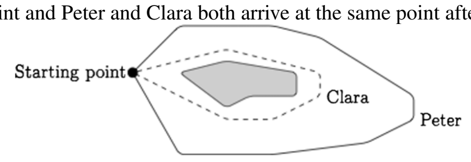
The second and third items, i.e., simple and larger instances, promote computer science. By solving a simple instance by hand, the student gets an insight on why a computer program is needed for solving the larger instances. While solving the small instance manually, the student also gets the logical intuition of the algorithm he is applying by hand. The answer to the simple instance, typically a single value, is used in the feedback to check whether the student has understood the purpose of the problem. In the very last question, the student has to provide an algorithm to solve the general problem.

Figure 1 shows an example of an isolated problem, as shown to the student. The algorithmic problem hidden behind the concrete situation is the least common multiple between two strictly positive integers.

Let's go for a tour around the lake

Context

Peter and Clara decided that they are going to go running around the lake. There are several possible paths around the lake. Peter and Clara both have their favourite paths. The two paths have the same starting point and Peter and Clara both arrive at the same point after having run.



Question

Let's suppose that Peter's path is **five** kilometres long and that Clara's one is only **three** kilometres long. If they start at the same time and if they are running at exactly the same speed, after **how many rounds** will Clara cross Peter **for the first time**?

Write a function that **takes two parameters** A and B which are **non-zero natural numbers** corresponding to the lengths of the paths of Peter and Clara. The function **returns a pair of natural numbers** containing the minimal number of rounds after which Peter and Clara (in that order) will cross each other at the starting point.

```
def toursNumber (A, B):
```

Fig. 1. The concrete situations introducing isolated problems motivate the students to complete the task. The algorithmic problem hidden behind the questions is finding the least common multiple.

Isolated problems are mainly targeted at people with some background in programming, but little knowledge of algorithm design. Students participating in mathematical olympiads are very interested in similar problems proposed by Project Euler. Such problems require the students to understand the problem, to be able to solve small instances by hand and then to build an algorithm out of their reasoning.

Small logic games used to increase algorithmic thinking (Burton, 2010) are well suited for isolated problems. Figure 2 shows an example of such tasks.

3.2. *Courses*

The Pythia framework aims to be a learning platform for people with zero programming knowledge or people wanting to improve their programming skills. We leverage the same platform used for isolated problems described in the previous section to provide full courses. The student can directly code what he is learning and gain feedback. A course in Pythia is a list of tasks. Unlike isolated problems, the student must complete the different

The mystery sequence

Context

In this problem, we are going to build a sequence of numbers. The first number of the sequence is the digit 1. To build the next numbers of the sequence, you have to execute the following procedure: you start from the previous number (in the sequence) and you read it from left to right, while counting the number of identical digits; you then indicate the obtained sum followed by the digit that was repeated and you do so until you have read the whole number.
Concretely, when you read a number from the sequence, digit by digit, it is in fact the description of the previous number in the sequence. To better understand, here are the first numbers of the sequence:

1. 1
2. 11
3. 21
4. 1211
5. 111221
6. 312211

Let's take the last number (312211). To build it, we start from the previous number (111221) and we read it from left to right. There is first **three** times the digit 1 (31), followed by **two** times the digit 2 (22) and finally **one** time the digit 1 (11).

Question

What is the 7th number of the sequence?

Write a function that **takes one parameter** N which is a **non-zero natural number**. The function **returns the** N**th number** of the mystery sequence.

```
def mysterySeq (N):
```

Fig. 2. Small logic games are good candidates for isolated problems and to learn programming and algorithm design.

tasks of a course in the given order, as they build upon each other. Courses are developed around the *learning by doing* motto (Dewey, 1938) which inspired active pedagogy learning methodologies.

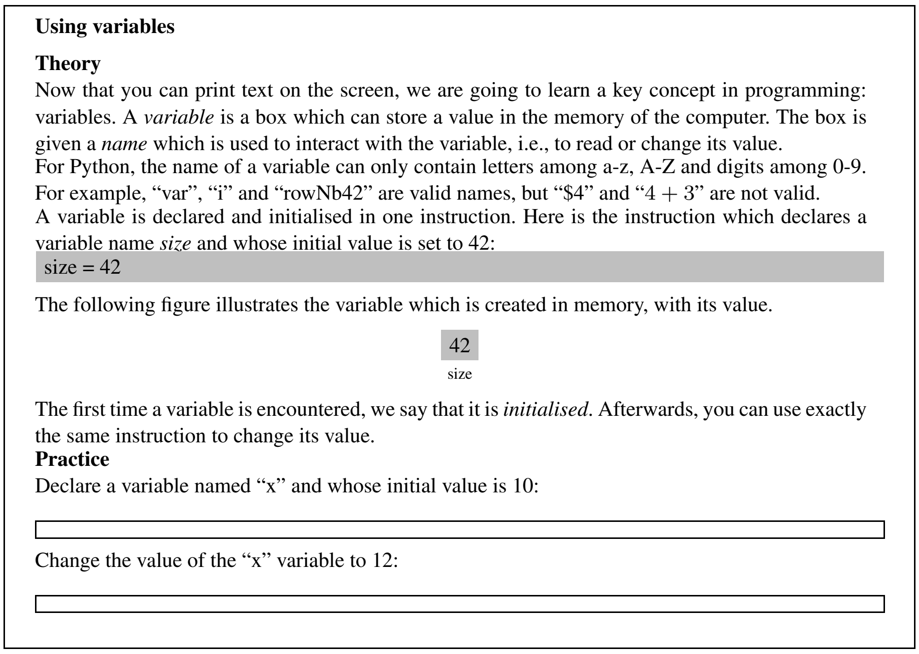
Figure 3 shows a task which belongs to the *Introduction to programming* course. The task introduces the concepts of variable, initialisation and value modification. Compared to isolated problems, context is replaced by theory and questions by practice. Theory is explained with illustrations and examples. Students can practise the new concepts on small exercises. For example, in Fig. 3 the student must declare and initialise one variable named "x" and then change its value.

3.3. *Feedback*

In order to keep a student motivated, and to help him/her learn something, he should receive feedback. With the proposed framework, the feedback can be directly provided to the student, as soon as his code has been executed. The framework is flexible enough to allow different kinds of feedback.

Using the capabilities of the Python language, the template program for the first isolated problem example presented in Fig. 1 is shown in Fig. 4. The tags `@@id@@` are placeholders for the student's answers. The tag `@    @q2@@` means that the student's answer tagged with the q2 id will be inserted with every line prefixed with four spaces.

The code for the tester is gathered in the `TestSuite` class. The class has the following methods.



Using variables

Theory

Now that you can print text on the screen, we are going to learn a key concept in programming: variables. A *variable* is a box which can store a value in the memory of the computer. The box is given a *name* which is used to interact with the variable, i.e., to read or change its value.
For Python, the name of a variable can only contain letters among a-z, A-Z and digits among 0-9. For example, "var", "i" and "rowNb42" are valid names, but "$4" and "4 + 3" are not valid.
A variable is declared and initialised in one instruction. Here is the instruction which declares a variable name *size* and whose initial value is set to 42:

```
size = 42
```

The following figure illustrates the variable which is created in memory, with its value.

42
size

The first time a variable is encountered, we say that it is *initialised*. Afterwards, you can use exactly the same instruction to change its value.

Practice

Declare a variable named "x" and whose initial value is 10:

Change the value of the "x" variable to 12:

Fig. 3. During a course, the students can immediately practise their newly learned skills on small exercises.

```
import random
def toursNumber(A, B):
@  @q2@@
class TestSuite:
  def correct(self, A, B):
    (x, y) = (A, B)
    if x == 0:
      return y
    while y != 0:
      if x > y:
        x -= y
      else:
        y -= x
    gcd = A * B / x
    return (gcd / A, gcd / B)
  def check(self, data, exp):
    try:
      ans = toursNumber(*data)
    except Exception as e:
      return 'exception:%s' % (str(e))
    if type(ans) is not tuple:
      return 'bad_type:%s:%s:%s' % (str(ans), str(data), str(exp))
    if len(answer) != 2:
      return 'bad_size:%s:%s:%s' % (str(ans), str(data), str(exp))
    try:
      a, b = (int(x) for x in answer)
      if a <= 0 or b <= 0:
        raise ValueError
    except ValueError:
      return 'bad_domain:%s:%s:%s' % (str(ans), str(data), str(exp))
    if ans != exp:
      return 'bad_answer:%s:%s:%s' % (str(ans), str(data), str(exp))
    else:
      return 'correct'
  def run(self):
    # Predefined tests
    inputs = [(1, 1), (3, 5), (1, 2), (2, 1)]
    outputs = [(1, 1), (5, 3), (2, 1), (1, 2)]
    correct = 0
    for i in range(len(inputs)):
      res = self.check(inputs[i], outputs[i])
      if res.split(':')[0] == 'correct':
        correct += 1
      else:
        print(res)
        break
    # Random tests
    if correct == len(inputs):
      nbtests = 100
      correct = 0
      for i in range(nbtests):
        A = random.randint(1, 100000)
        B = random.randint(1, 100000)
        ans = self.correct(A, B)
        res = self.check((A, B), ans)
        if res.split(':')[0] == 'correct':
          correct += 1
        else:
          print(res)
          break
      if correct == nbtests:
        print('correct')
TestSuite().run()
```

Fig. 4. Placeholders in the template program will be replaced by the student's answers. The completed program is run in a sandboxed environment.

- The `correct` method contains the reference code with correct implementation.
- The `check` method executes the student's code on one data test. It returns a string code indicating the kind of error and that will be used in the analyser script.
- The `run` method runs the whole test suite. A test suite is composed of a set of predefined tests (to force testing some limit cases) and of a set of random tests (to disallow the student to hardcode the answers in his/her solution).

Note that, as the program is executed inside a sandbox with no contact to the outside world, it cannot provide direct feedback to the student. Instead, the program prints out the results to be analysed by the analyser script.

The template program just shown in Fig. 4 is a standard test suite program. The flexibility of Pythia makes it possible to have richer kind of feedbacks. In the task shown in Fig. 5, the student is required to implement a search algorithm on an array of natural numbers. The expected solution is a dichotomic search. The parameter stack that the student receives is an array-like object whose [] method has been overridden so as to be able to count the number of times the array is accessed. The provided feedback then includes a plot of the number of accesses compared to the size of the problem (i.e., the size of the array). The plot graphically shows to the student the time complexity of his algorithm, compared to the expected complexity. The plot is generated by executing the student's code and the reference code on arrays with increasing size. Outputs generated by the template program contains the number of array accesses and are used by the analyser script to generate the plot with Google Chart Tools.

The distinction between the template program and the analyser script opens a lot of opportunities.

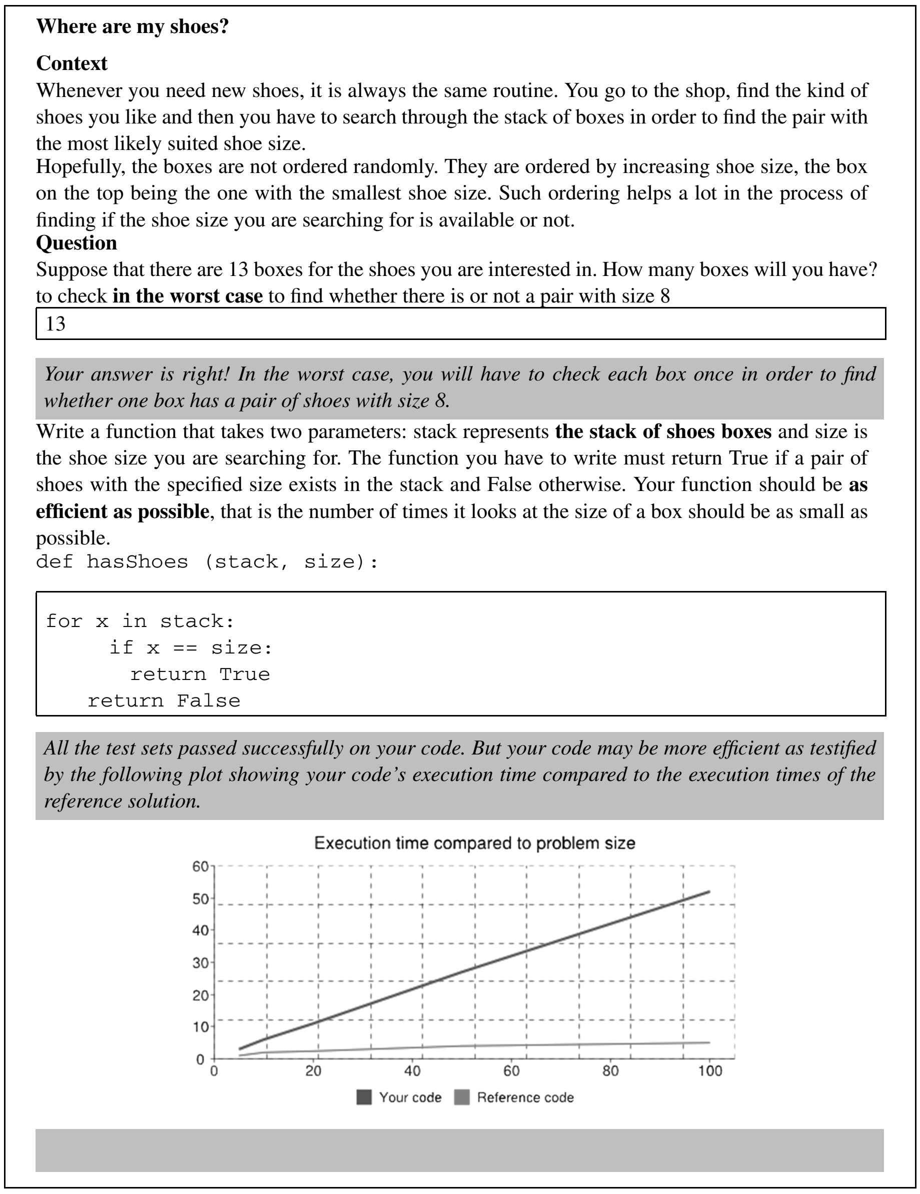
The teacher can also use feedback to grade students, e.g., in the context of using Pythia for homework. Grading is achieved by customising the analyser script to generate private information for the teacher alongside the public information dedicated to the student. The two parts can be linked to a private and a public test set respectively, defined in the template program.

Moreover, by separating the analyser script, it is possible for a teacher to support multilingual courses and provide feedbacks in multiple languages (which is useful in Belgium which has three national languages). The multi-language support is managed in the analyser script.

For now, those customisation are not easy to realise since it should be done in the analyser script by the teacher. Future work includes developing a framework to ease the task of writing template programmes and analyser scripts for teachers.

4. The Pythia Framework

This section describes the global architecture of the Pythia framework. The framework allows students to solve tasks and get useful feedback about their submissions. Such feedback involves running user-submitted code. Special care has been taken in Pythia to handle the risks of executing buggy or malicious programs. Please note that the framework is



Where are my shoes?

Context

Whenever you need new shoes, it is always the same routine. You go to the shop, find the kind of shoes you like and then you have to search through the stack of boxes in order to find the pair with the most likely suited shoe size.
Hopefully, the boxes are not ordered randomly. They are ordered by increasing shoe size, the box on the top being the one with the smallest shoe size. Such ordering helps a lot in the process of finding if the shoe size you are searching for is available or not.

Question

Suppose that there are 13 boxes for the shoes you are interested in. How many boxes will you have? to check **in the worst case** to find whether there is or not a pair with size 8

13

Your answer is right! In the worst case, you will have to check each box once in order to find whether one box has a pair of shoes with size 8.

Write a function that takes two parameters: stack represents **the stack of shoes boxes** and size is the shoe size you are searching for. The function you have to write must return True if a pair of shoes with the specified size exists in the stack and False otherwise. Your function should be **as efficient as possible**, that is the number of times it looks at the size of a box should be as small as possible.

```
def hasShoes (stack, size):
```

```
for x in stack:
     if x == size:
       return True
   return False
```

All the test sets passed successfully on your code. But your code may be more efficient as testified by the following plot showing your code's execution time compared to the execution times of the reference solution.

Fig. 5. The feedback mechanism is very versatile. For this problem, the student is presented a feedback with a plot representing the time complexity of his algorithm in a graphical way. The expected solution, a dichotomic search, has logarithmic time complexity.

language-agnostic. Problems can be written in different programming languages as long as an interpreter or compiler is available. Figure 6 illustrates the different components of the framework and the relations between them.

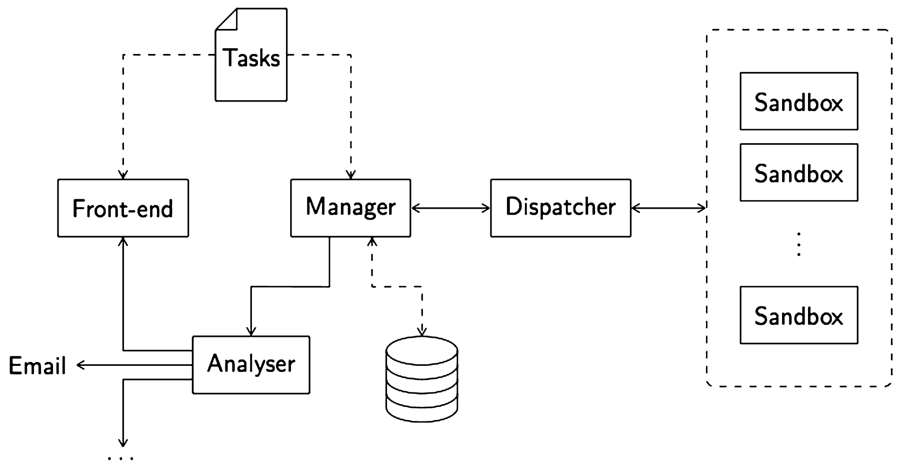


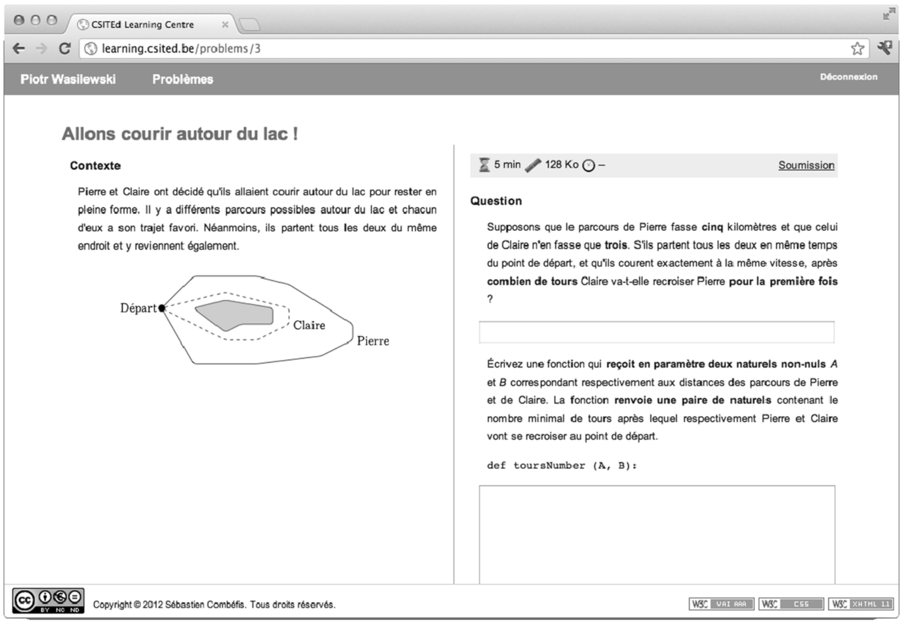
Fig. 6. The Pythia framework is subdivided into separate components communicating with each other with simple HTTP requests. Plain arrows represent communication between components and dashed arrows represent information access.

The programming courses and the problems are defined as *task descriptions* as shown in the previous section. Explanation and questions are presented to the student through the *front-end*. Using the front-end web application, the student can submit answers to the questions. Answers are used by the *manager* to fill in the placeholders of the template program. The resulting complete program is executed in a safe environment and its outcome is analysed by the analyser script. The remainder of this section dives more into the inner working of the system.

The manager, dispatcher and sandboxes are all small web servers communicating with each other with simple HTTP requests. Currently, we assume the existence of a shared filesystem between all components. Such assumption holds when all components are run on the same machine or if a networked filesystem is used.

The *manager* serves two roles. When it receives a submission from the frond-end, the manager merges the template program with the student's answers. The complete program, called a *job*, is then sent to the dispatcher. Later, when the job's execution has completed, the manager launches the *analyser* script. The analyser is responsible for providing feedback, based on the results of the computation. Feedback to the student can be given through the front-end web interface, by email, or by other means. The analyser can also provide the student's grade to the teacher. In order to provide statistical information to the users and to follow them, the manager stores information in a database.

Jobs are executed in *sandboxes* due to security concerns posed by buggy or malicious code. A sandbox is a disposable virtual machine that is created specifically for a job. Full virtual machines that emulate hardware peripherals, like KVM or VirtualBox, are not suitable for such task as they are too slow to start up. Instead, Pythia uses User-mode Linux (Dike, 2000), a port of the Linux kernel implementing drivers by forwarding the requests to the host Linux kernel. The sandbox runs a heavily trimmed down version of ArchLinux and is able to boot in under one second. The root filesystem being read-only,



Allons courir autour du lac !

Contexte

Pierre et Claire ont décidé qu'ils allaient courir autour du lac pour rester en pleine forme. Il y a différents parcours possibles autour du lac et chacun d'eux a son trajet favori. Néanmoins, ils partent tous les deux du même endroit et y reviennent également.

Question

Supposons que le parcours de Pierre fasse **cinq** kilomètres et que celui de Claire n'en fasse que **trois**. S'ils partent tous les deux en même temps du point de départ, et qu'ils courent exactement à la même vitesse, après **combien de tours** Claire va-t-elle recroiser Pierre **pour la première fois** ?

Écrivez une fonction qui **reçoit en paramètre deux naturels non-nuls** *A* et *B* correspondant respectivement aux distances des parcours de Pierre et de Claire. La fonction **renvoie une paire de naturels** contenant le nombre minimal de tours après lequel respectivement Pierre et Claire vont se recroiser au point de départ.

```
def toursNumber (A, B):
```

Copyright © 2012 Sébastien Combéfis. Tous droits réservés.

Fig. 7. The students complete tasks using the web-based front-end.

the job's output is written to a RAM-backed temporary filesystem. As such, the total amount of main and storage memory a job can use is capped by the amount of memory that is made available to the sandbox kernel. At the end of the job execution, the output files are copied back to the host in the shared filesystem.

Finally, the *dispatcher* makes the link between the manager and the sandboxes. The dispatcher maintains a queue of jobs to be run and dispatches them to idle sandboxes, which may or may not be on the same machine. Once a job is finished, the dispatcher notifies the manager.

5. Evaluation

An implementation of the Pythia framework is currently under development. Figure 7 shows the front-end of the *"Let's go for a tour around the lake"* problem. Once finalised, the working prototype will be used to evaluate our approach in two steps.

During the first step, a small group will be acting as beta-testers for both the framework itself and the courses and problems available on the platform. The group will consist of about ten persons, mostly bachelor students in computer science and in engineering but also older people with very few programming knowledge.

The second step will be to send invitations to secondary school pupils (12–18 years old) who are the main target audience of Pythia.

6. Conclusion

To promote computer science and to answer the needs highlighted by the contestants of the Belgian Olympiad in Informatics, the Pythia framework, a web-based learning platform, is being developed.

The framework proposes a set of programming courses and isolated problems. Students' programs submitted as answers are safely executed and the students receive fully automated feedback. The proposed problems are designed with the students' motivation in mind. Following the philosophy of learning by doing, the tasks aim at keeping the students interested in continuing and learning.

A prototype has been developed to evaluate our approach.

Future work includes strengthening the framework, creating more tasks and courses, supporting more programming languages (we currently target Python), thinking about the internationalisation of the front-end (useful in a multi-languages country such as Belgium). Future work also includes extending the framework to ease the task of writing a problem for the teacher, that is the description, the template programme and the analyser script. In the long run, the framework should also be tested in other contexts such as automated homework grading or online contests. In such applications, extensions like cheating detection could be considered.

References

Bell, T., Alexander, J., Freeman, I., Grimley, M. (2009). Computer science unplugged: school students doing real computing without computers. *Journal of Applied Computing and Information*, 13(1), 20–29.

Burton, B. (2010). Encouraging algorithmic thinking without a computer. *Olympiads in Informatics*, 4, 3–14.

Codecademy. `http://www.codecademy.com`.

Combéfis, S., Leroy, D. (2011). Belgian olympiads in informatics: the story of launching a national contest. *Olympiads in Informatics*, 5, 131–139.

Dewey, J. (1938). *Experience and Education*. New York, The Macmillan Publishing Company.

Dike, J. (2000). A user-mode port of the Linux kernel. In: *Proceedings of the 4th Annual Linux Showcase & Conference*, Atlanta, GA, Usenix.

hackzor. `http://code.google.com/p/hackzor/`.

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., Resnick, M. (2004). Scratch: a sneak preview. In: *Proceedings of the 2nd International Conference on Creating, Connecting, and Collaborating through Computing*, Kyoto, Japan, 104–109.

Mareš, M. (2009). Moe – design of a modular grading system. *Olympiads in Informatics*, 3, 60–66.

Project Euler. `http://projecteuler.net`.

Rubymonk. `http://rubymonk.com`.

Try Python. `http://www.trypython.org`.

Try Ruby. `http://tryruby.org`.

uevalrun. `http://code.google.com/p/pts-mini-gpl/wiki/uevalrun`.

Viau, R. (1998). *La Motivation en Contexte Scolaire*. Édition de Boeck (2ème édition), Bruxelles.

S. Combéfis is a PhD Student at the Université catholique de Louvain in Belgium and works as a teaching assistant for the Computer Science Engineering Department. He is also following an advanced master in pedagogy in higher education. In 2010, he founded, with Damien Leroy, the Belgian Olympiads in Informatics (be-OI). He is now part of the coordinating committee that is in charge of managing everything which is related to the national contest. He is also trainer for the Belgian delegation to the IOI.

V. le Clément de Saint-Marcq is a PhD student at the Université catholique de Louvain in Belgium, funded as a research assistant by the FNRS, the national fund for scientific research. He works for the Computing Science Engineering Division of the ICTEAM Institute. Together with Sébastien Combéfis, he is involved in training the Belgian delegation to the IOI.