Pythia Reloaded: An Intelligent Unit Testing-Based Code Grader for Education

Dr **Sébastien Combéfis**[1,2] Alexis Paques[2]

[1]École Centrale des Arts et Métiers (ECAM)

[2]Computer Science and IT in Education (CSITEd)

July 14, 2015

[CHESE 2015, Baltimore, MD, USA]

Context

- Automatic assessment of codes

  Programming learning platforms, MOOCs,
  higher education courses, competitions...

- Two major kinds of code assessment
  - Unit testing frameworks
  - Competition graders

- Platform for automatic assessment and "intelligent" feedback

  Suited for education and that helps learners

Pythia framework

- Released as an open-source project

  http://www.pythia-project.org

- A unit testing framework...

  Functional tests on learners' code

  ...and a competition grader

  Sandboxed environment with execution constraints

- Prototyped in 2012, used for a MOOC in Spring 2014

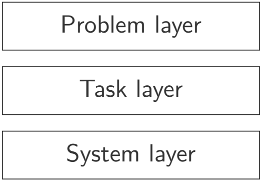
Architecture

- The Pythia framework consists in three layers

System layer Safe execution of jobs

Task layer Feedbacks

Problem layer Context and input/output structure



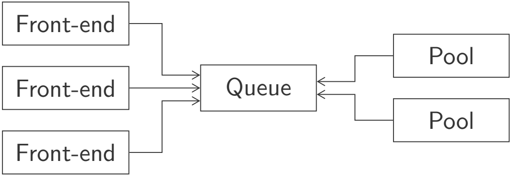
Problem layer

Task layer

System layer

System layer

- Front-ends communicate with the outside world
- Pools manage the safe execution of jobs



Job execution

- Sandbox uses UML with trimmed-down version of ArchLinux

  The VM boots in under one second

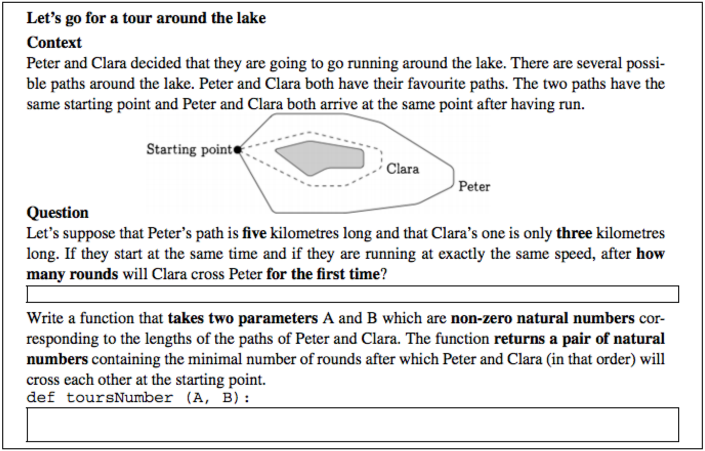
- A job has an environment and a task filesystem

  Constraints can be added to the VM (time, memory and output)

```
{
  "environment":  "python",
  "taskfs":  "hello-world-python.sfs",
  "limits":  {
    "time":  60,
    "memory":  32,
    "disk":  50,
    "output":  1024
  }
}
```

Problem example

- A problem combines several tasks and adds a context



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.

Starting point
Clara
Peter

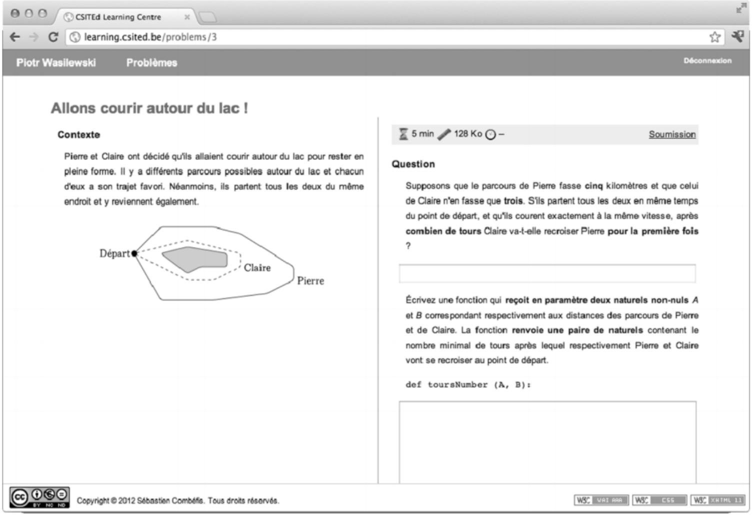
Question

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):
```

Pythia LMS



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):
```

Unit Testing-Based Grading

- Different kinds of tasks are possible

  Examine stdout, evaluate time complexity, run unit tests...

- Unit testing is for developers

  Feedback is related to the specification of the function

- Feedback for learners is related to the problem to be solved

Main executable I

1. **Preprocess**

   Integrates code snippets from learners into skeletons

2. **Compile**

   Analyses statically the code and compiles it

3. **Generate**

   Generates tests sets and saves them to a file

Main executable II

4. **Execute**

   Executes learner's code against tests sets, generates data

5. **Postprocess**

   Analyses generated data, and produces analysis results

6. **Feedback**

   Generates feedbacks

Configuration-based definition

```
{
  "q1":  {
    "argc":  2,
    "predefined":  {
      "argv":  [{
        "data":  "(10, 5)",
        "feedback":  {
          "10":  "Have you summed the 2nd parameter?",
          "5":  "Have you summed the 1st parameter?"
        }
      }, {
        "data":  "(7, 15)"
      }, {
        "data":  "(-1, 2)",
        "feedback":  {
          "*":  "Have considered negative parameter?",
        }
      }, {
        "data":  "(12, 0)"
      }]
    },
    "random":  {
      "n":  10,
      "args":  ["int(-20,20)", "int(-20,20)"]
    },
    "code":  "def sum(a, b):\n return a + b"
  }
}
```

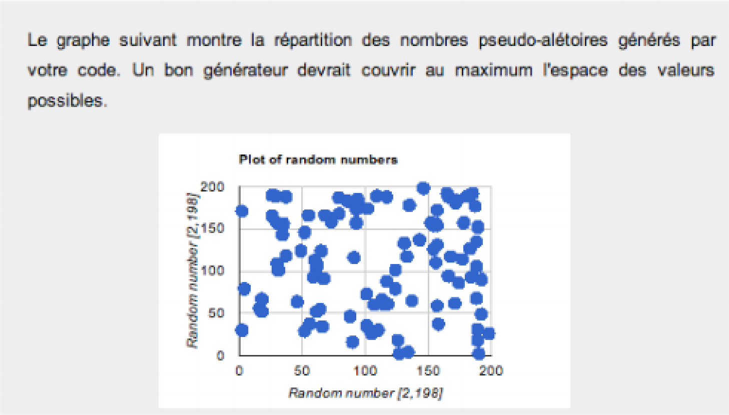
Importance of feedback

- Predefined tests must cover errors often made by learners
- Feedback should be related to the problem being solved

"Your code failed for the input $a = 10$, $b = 5$.
The expected result is 15 and your code produced 10.
Have you summed the 2nd parameter?"

Visual feedback

- Visual feedback to be interpreted by the learner



Le graphe suivant montre la répartition des nombres pseudo-alétoires générés par votre code. Un bon générateur devrait couvrir au maximum l'espace des valeurs possibles.

Conclusion

- Pythia is an open-source platform containing a unit testing-based grader specifically designed for education
- **Ongoing work**
  - Definition and specification of new kinds of tasks
  - Development of Pythia Studio and Pythia LMS
- **Future work**
  - Explore use of other kinds of code checks
  - Mining code to identify common errors to extract feedback