Olympiads in Informatics, 2012, Vol. 6, 205–211

Putka – A Web Application in Support of Computer Programming Education

Jelko URBANČIČ[1], Mitja TRAMPUŠ[2]
[1]*Zavod za računalniško izobraževanje Ljubljana (Institute for Computer Education Ljubljana)*
p.p. 4716, SI-1001 Ljubljana, Slovenia
[2]*Jozef Stefan Institute*
Jamova 39, SI-1000 Ljubljana, Slovenia
e-mail: jelko.urbancic@guest.arnes.si, t.mitja@gmail.com

Abstract. In many countries the interest in pre-university computer programming education has been changed during the last decade. In the Slovenian case, the education reform, demographic and other reasons produced a big decrease of interest among young people. The paper describes our approach to increasing the interest for computer programming by integrating a web based tool "Putka" into the education process. Putka is, at its core, an online judge. Two facts make it stand apart from similar applications, however. First, its design and many additional features are influenced by its heavy use in support during long-term programming classes rather than only one-off competitions. Second, while its use in the classroom was the primary motivation for developing Putka, it has also hosted numerous competitions with varying rules. Out of necessity, this multifaceted nature bore what we believe to be a very flexible technical design.

Key words: computer programming education, web based education tool, online judge.

1. Introduction and Historical Background

The Slovenian story about the attitude of young population to the informatics is similar to that from some other developed countries. In the first period, during the introduction of computers into homes at the end of the 80s and the beginning of the 90s we had a very strong interest. The result of this boom was quite a large number of primary and secondary school contestants, 3000 in the country of two million inhabitants.

Computer education was used as a synonym for computer programming. The education was lead mostly as additional courses in primary and secondary schools and the rest was lead by private enterprise and the civil society movement. One organization is of particular interest for our story: Zavod za računalniško izobraževanje Ljubljana (abbr. ZRI, eng. Institute of computer education Ljubljana), a private organization founded in 1989. The institute expanded its activity fast and the number of primary and secondary school students rose up to 1000 by 1995. Remarkably, the Ministry of Education had no special strategy or program for the introduction of computers and informatics in primary schools and a very weak one for the secondary schools until 1992.

The second period, starting in 1995, saw a significant decrease of interest for computer programming among the young, similarly to other developed countries.

Besides all sociological, demographic and other reasons, the Ministry of Education affected the decline significantly. A large amount of money was spent in computer technology. Most of the primary schools in the country built a computer classroom. But the education program through which the computer classroom should be put to use was weak and the competence of the majority teachers was insufficient. A big loss of motivation for informatics and computer programming consequently occurred among young people. The number of young students taking computer programming classes and the number of contestants were both reduced by at least by factor of five in a few years. All private and civil society organizations either disappeared or switched to other business except ZRI, but it too was forced to decrease the number of students down to 250 by 2002.

The third and current period was marked by the introduction of another school reform that caused shock to many extracurricular activities including computer programming. Part of the reform is the introduction of elective courses for primary schools. While this has its benefits, the list of available courses is in reality often severely limited by staff and budget resources at individual schools. A key factor in deciding which courses to offer is the students' interest; computer programming as a somewhat fringe activity has, to our knowledge, never been realized as an elective course. On the other hand, the children each got their own schedule, causing them to be fragmented, with school courses finishing late in the afternoon.

The daily time frame for all children's out-of-school activities was mostly reduced to a two-hour interval between five and seven in the evening. As the demand for extracurricular programming courses shrank, there was no more room for full-time programming teachers. As a consequence, the quality dropped: a teacher who also teaches unrelated subjects cannot dedicate enough time to properly prepare for a programming course.

Many civil society organizations and private institutions for children out-of-school activities collapsed. The remaining ones faced stronger competition from all kinds of activities.

This was a very strong shock for computer programming. The number of contestants fell from 3000 in the first half of the 90s to less than 100 nation-wide in the period from 2005 till 2010, finally climbing to 200 in 2012.

All this time, ZRI was a leading institution in the field of computer programming education for the young population. It changed its business model in 2002 and at first only made its courses, taught by voluntary teachers, available to only about 20 most motivated students. The number of students has then been increased step by step to the present number of about 40. This is a part of our mission of preserving the organizational and methodological knowledge of teaching the young population computer programming. The second part of ZRI's vision is to prepare more or less competent contestants for the IOI (Urbančič, 2008). At the time when the number of young students practicing computer programming was at its minimum we were thinking about what we can change. We can influence neither the demographic factors nor the educational policy within the country. The students' motivation is the only factor that can be influenced by us. We believed a user friendly learning tool that provided students with instant feedback and progress reports would enhance their motivation. Although online judges with ample task collections already existed, very few of them met even one out of our three criteria:

- a large number of entry-level tasks systematically covering the basic programming concepts, e.g., simple loops or conditionals;
- tasks and user interface in Slovene (important for young students);
- the possibility of administering the system: creating user groups, tracking progress of students, creating tasks and possibly contests . . .

Surprisingly, this is still largely true today. Apart from the obvious Slovene language barrier, most judges (UVA toolkit, 2012) are not publically available as a software package and mostly support only ACM-style problems (fixed input, fixed output). We therefore decided to build a tool of our own as an institute project. The web was deemed to be the most suitable platform.

During our first attempt in 1999, we seriously underestimated the complexity of the task at hand: not only did we have little experience with both project management and web development; we also did not start off with a clearly defined idea of what the end result of the project should be. Once we realized all this, we cancelled the project, postponing the general idea of developing a web-supported learning platform for later.

During our next attempt, in 2005, we used an approach influenced by a previous project together with some good practices of project management. This resulted in a well-rounded program which was however still plagued by instabilities, the result of our overly-eager experimentation with new technologies.

Finally, the third attempt resulted in the first operational version and since July 2007 the application is available at www.putka.si.

2. Description of the Present System

Putka grew along with the scope at which it was being applied. At first designed to support classroom courses in a very basic manner, it got expanded over time with **numerous convenience features**, e.g., task search, tagging, difficulty ratings, task grouping and so on. While the improvements are mostly simple, they are the result of real needs identified during years of experience and therefore significantly improve the teaching experience.

In addition to the classroom setting, Putka soon supported and got used for organizing competitions. To date, it hosted several dozen competitions (including at national level; Putka, 2012), both on-site and online, with **varying rules**, among them ACM (ICPC) (ACM Putka, 2012) and IOI. Competitions bring not only diversity in rules, but also in **types of tasks**. We support classic input-output tasks, tasks with non-unique outputs, interactive tasks, tasks with manually-graded outputs and likely even scenarios we failed to foresee. This is due to an extremely flexible (but succinct) *test script* framework for specifying evaluation procedures (see Section 2.3).

2.1. *System Architecture*

In supporting this large set of use cases, a highly modular design was instrumental. There are four main components to Putka:

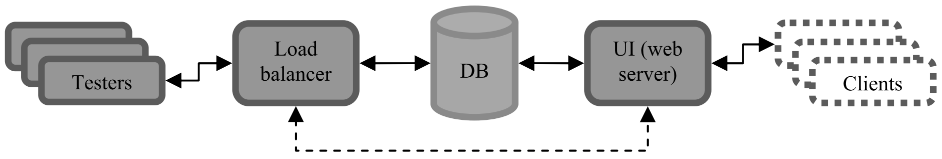


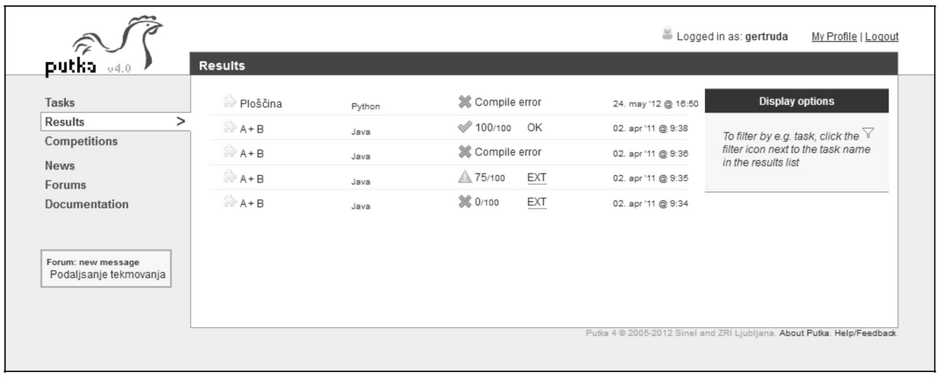
Fig. 1. The communications and data flow diagram between the main system components. The dashed line represents notifications of DB changes.

- *the database* – the single point of persistent data storage;
- *the manager* – distributes work among testers (load balancer);
- *the tester(s)* – evaluate user programs in a strictly controlled environment;
- *the user interface* – a web application.

Each of the components can be run on a separate computer. In practice, we tend to run everything but the testers on a single machine. The communication between the components is kept as streamlined as possible. Figure 1 illustrates the communication links.

2.2. *User Interface*

The user interface is realized as a web application, allowing students access to their learning environment from virtually anywhere.



Ploščina	Python	Compile error		24. may '12 @ 16:50
A + B	Java	100/100	OK	02. apr '11 @ 9:38
A + B	Java	Compile error		02. apr '11 @ 9:36
A + B	Java	75/100	EXT	02. apr '11 @ 9:35
A + B	Java	0/100	EXT	02. apr '11 @ 9:34

To filter by e.g. task, click the filter icon next to the task name in the results list

Forum: new message
Podaljsanje tekmovanja

Putka 4 © 2005-2012 Sinel and ZRI Ljubljana. About Putka. Help/Feedback.

Fig. 2. A screenshot of the user interface showing the overview of user's submissions. The administrative/teacher's part of the UI is not shown.

Features. The UI consists of several modules, accessible through the main menu. Of central importance in Putka are the programming tasks. They are hierarchically organized into directories equipped with a full ACL (Access Control List) permission system, allowing fine-grained control over which groups of students can view or edit[1] which tasks.

[1]Some senior students are actually editors/authors of simpler problems (a valuable experience), but their access to the hard problems should clearly be limited.

A separate module shows aggregate and detailed upload results; this can be a very lively and informative view for administrators during a competition or a class session. The tasks and results view(s) are tightly interlinked and support various searches and filters – with hundreds of tasks and users, this is essential especially for the teachers. Competitions are managed in a small separate module. Forum is the last of the modules involving tasks relatively strongly. It features a general-purpose online forum with additional automatically created discussion threads about individual tasks. These threads are also linked to from the task pages and students involved with a task receive a notification on new posts; this is particularly useful during competitions when the forum is the only means of communication.

Users and their groups can be managed in a separate module. Administrators can get an overview of each user and track his progress by using this module. The **documentation** module hosts reference manuals and tutorials for several programming languages. All the pages are mirrored (hosted locally). This self-contained setting is crucial for on-site competitions with no internet allowed.

Technology. The UI is written in Django. We make use of Django's object model to keep the data cleanly organized and structured. We also tightly integrated the data model with a server-push framework running on top of Tornado/Tornadio, a non-blocking web server. This allows for pushing notifications to the web browser (e.g., ”a task has finished its waiting/evaluation”, ”a competition has started”, ”new unread message” . . .) without the user having to reload the page. A relatively small part of the UI code also runs on the client site, written in javascript. However, we made it a point to keep the interface usable, if less comfortable, even with javascript disabled.

2.3. *The Testing Backend*

By the backend, we refer to the part of Putka concerned with evaluating users' programs in a controlled environment. The backend is controlled by a single **manager**, an application that distributes work among **testers** and provides a low-level monitoring interface to them. The testers compile and run the user programs. This is performed in a **debugger-like** fashion: all system calls made by the user programs are intercepted and blocked or allowed based on the programming language (e.g., the python interpreter needs access to /etc/passwd) and/or the task. This prevents almost all forms of attacks on the backend. The time and memory consumption of user programs are of course also being monitored and limited.

The testers run on machines with different hardware configurations, the time limits get adjusted based on a ”**speed factor**” computed for each computer. However, we have learned to be wary of such situations: it turns out that the speed ratio between two machines can **vary wildly** (5x and more) depending on the program being evaluated and its speed bottleneck (RAM, 64-bit operations . . .).

Test scripts: Defining the Test Scenario. The tester is a C++ application which **embeds the python interpreter** and exposes to it some basic Putka-related functions (configuring the jail, executing programs in the ”debugger”, reporting results to the manager).

On top of these, a python layer provides **convenience functions** encoding the most common test scenarios (e.g., ”compile, test on all input-output pairs, and sum the scores of test cases”). The evaluation procedure and scoring are fully defined by a python *test script* that is a part of each task and gets interpreted in this environment. The convenience functions enable the scripts to be very short in most cases (often a single function call), while the low-level functions and the power of the full python interpreter keep almost any conceivable test scenario expressible in our framework.

3. The Present Role of the Putka System

The mission of the Putka system is to motivate students and improve the quality of the courses on one hand and to provide a quality contest environment on the other. Very importantly, the system needs to be offered in the national language.

To this end, today the system is managed by the Putka team[2] and is widely used for courses at ZRI. An archive of around 300 problems is available to the students. The problems are only gradually revealed to the students according to their progress. The students can upload their solution from anywhere (ZRI, home etc.). Each student can access his archive of solutions. The coaches have access to student’s archives to monitor their progress and to facilitate problem discussion. There is also a module for online discussion, used outside the classroom and during competitions.

The required competences of the users are set very low to include the Putka system in the **early education process** as soon as possible. Students must have basic knowledge of standard input-output operations and fundamentals like conditionals and loops – many tasks are formulated only using these simple constructs. The very easiest tasks do not even require loops; beginners aged 12-16 are normally introduced to Putka after ten hours of introductory courses.

To promote programming among the young, the whole system with a limited set of exercises is publicly available and anyone can access it upon registering at www.putka.si.

Besides the learning/classroom mode, the contest mode is being actively used as well, with multiple internal and national contests using IOI and ACM rules being executed each year.

4. Conclusion

From 2005/06 when the number of students attending ZRI and/or computer programming contests in Slovenia reached its minimum, the attendance almost doubled by 2012. There were some good promotional activities in Slovenia lead by the ACM in these years, but at least for the turn for the better at ZRI, the impact of systematic approach using the Putka system was crucial. The system’s design and choice of features, both based on previous teaching experience, have been proven to be successful through five years of continuous use in the classroom and at competitions.

[2]A volunteer group of ZRI teachers and (former) students.

References

ACM Putka (2012). Available at: `http://putka.upm.si/tasks/`.
Putka (2012). Available at: `http://www.putka.si/`.
Urbančič, J. (2008). 20 mednarodna računalniška olimpijada, Presek, Društvo matematikov, fizikov in astronomov Slovenije, 36(2), 27–29.
UVA Toolkit (2012). Avaliable at: `http://uvatoolkit.com/links.php`.

J. Urbančič had been involved in computer programming at eighteen. He received PhD in atmospheric sciences (1988), two years later founded ZRI where he has been acting director until 2002. Since then, he has been the manager in the state administration and besides that he is volunteering at ZRI and national association as a part time teacher, coach, judge and organizer of the competitions. He attended IOI between 2006 and 2010 as deputy leader of national team.

M. Trampuš has been actively involved with computer programming education since he was 10 – first as a student and competitor and national and international levels (CEOI, IOI, CERC), later as an organizer and occasional teacher. Recently, he attended CEOI 2011 and 2012 as the national team leader. He is one of the three core software developers of the present Putka system. Currently, he is working on his PhD in the area of data mining.

From the beginning the **Putka team** has had following active members: Nino Bašič, Jan Berčič, Žiga Ham, Tomaž Hočevar, Mitja Trampuš, Jelko Urbančič and Andrej Veber. In previous versions Andrej Bukošek, Domen Blenkuš and Peter Koželj were also active. Recently, three new young colleagues joined the team. Presently nine members of the team cover the roles of system administration, softwaredevelopment, task administration and user administration. The name Putka is the acronym of "Verifying the effectiveness, thoroughness and correctness of algorithms" in Slovene language. It is also the expression for a lovely little hen, that's why it has a hen in its logo.