Reuse of educational tasks that support e-learning automation

Ilya Posov

Educational tasks are used by teachers and students to monitor and evaluate their learning. The features of tasks may vary a lot, for example, tasks may contain some or all of the following: a text with a statement, an answer, automatically checked answers, feedback for student responses, hints, generation of statements, interactive interface, etc. Most of these features need specialized learning software, and usually tasks prepared for one learning system are not supported by other systems. If a teacher has a large tasks repository in one learning system, that does not fully fit his or her needs, it is usually impossible to reuse this repository in another learning system. This may lead to usage of several learning systems with different tasks sets.

Ideally, a teacher has one tasks repository and a set of software tools used in different educational situations. Each tool supports a subset of tasks, and the more intersections these subsets have, the more useful actions a teacher may perform with an average task.

This may be achieved if we consider a task as an object in terms of the object oriented paradigm. A task exposes several interfaces that it implements. For example, a task may have actions to get its statement for displaying to a student, to grade a student's response, etc. Many e-learning automation routines become a responsibility of a task, and not of a learning system. For example, the generation of a statement is done by a task when a learning system asks for its statement; an answer grading is done by a task by either comparing the answer with a hard-coded correct one, or by running a computer algebra system with checking computations. The task is represented as a structured data in a file or a byte stream, with the information about exposed interfaces, so that a learning system may discover task's abilities and decide, whether it supports a task or not. The implementation of interfaces may be done inside of the problem, but tasks usually exists as sets of similar tasks, so the implementation may move out of the task to not repeat itself.

The tasks and learning systems, that use them, are not the only objects in the proposed architecture. One also need adapters that implement interfaces by means

of other interfaces. For example, tasks from a bebras contest in informatics have many information slots such as a statement, a question, a list of possible answers, an explanation, definitions of concepts, connection with informatics, etc. These are exposed as an interface of a task for a bebras contest, and only the bebras contest system may understand all this information. But if we have an adapter that converts this interface to the interface that allows to get tasks statement, we may use bebras problems in almost any learning system, because they will be able to show their statement to students, that is sometimes enough.

The proposed architecture is similar to audio and video processing software: there are many processing tools, there are many audio and video files of different formats, and there are codecs that make these formats understandable for processing tools.

The implementation of a proposed architecture needs to address many problems, that are well-known because the idea of stand-alone object, i.e. objects that exist outside of the software, is not new. Some of them are about a setup of an environment, security, performance: a learning system may not expect that getting a statement sometimes takes too much time because this operation needs an access to a computer algebra system. The report presents an overview of the architecture with the discussion of how to solve the problems and make the system usable for at least the types of tasks that the author works with. They include tasks for different types of online and offline contests, research laboratories, tasks for students for different school and university courses (mathematics and programming in general purpose programming languages and mathematical software), many of task types are generable and suppose automatic grading.

Ilya Posov
Faculty of Arts
St. Petersburg State University,
Saint-Petersburg, Russia,

Faculty of Computer Science and Technology
St. Petersburg Electrotechnical University "LETI"
Saint-Petersburg, Russia
e-mail: i.posov@spbu.ru
e-mail: iposov@gmail.com