A SOA-Based Framework for Constructing Problem Solving Environments

Jaime Gálvez, Eduardo Guzmán, and Ricardo Conejo
Dpto. de Lenguajes y Ciencias de la Computación, Universidad de Málaga.
Bulevar Louis Pasteur, 35. Campus de Teatinos. Málaga, Spain
{jgalvez, guzman, conejo}@lcc.uma.es

Abstract

In this paper we present a framework for constructing problem solving environments for assessing procedural knowledge, i.e. the student's ability to apply his/her knowledge in order to accomplish a task. Our proposal combines the most recent technologies for web-based development (e.g. service oriented architectures, JSF, JBoss rules, etc.) with a well-founded theory to make sound student knowledge estimations and to carry out diagnosis adaptively.

1. Introduction

Assessment could be defined as the process of inferring what a student knows, based on evidence obtained from the student's actions or behavioural observations [3]. The relevance of assessment is very high, since it allows observers to determine whether or not the student has assimilated the notions introduced during a learning process.

In this paper we present a web-based framework for constructing problem solving assessment environments. This framework has been developed as a Service Oriented Architecture (SOA) [1] and, accordingly, its functionalities have been organized in a set of services. Our main goal is to facilitate the construction of problem solving assessment tools and also to provide these systems with mechanisms for making the assessment process adaptive and/or to guarantee well-founded diagnosis of student's knowledge.

Using this framework we have developed a problem solving tool to evaluate whether the students are able or not to apply the Simplex or the Two-Phase algorithms in order to solve linear optimization problems.

The paper is structured as follows. In section 2 we describe the architecture of our framework and the services it provides. In section 3 we approach the Simplex assessment tutor, developed using the framework. Finally, we comment on the contribution of this paper and outline future work.

2. Architecture

Our proposal is a web-based and service-oriented framework designed on the J2EE platform. Accordingly this framework takes advantage of the services and facilities this technology offers.

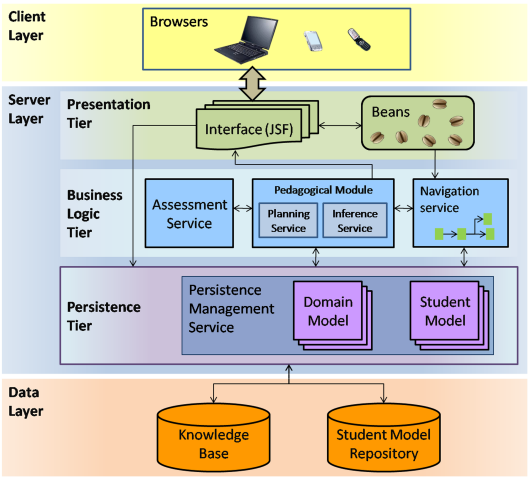

Figure 1. Tutor Architecture.

We distinguish three layers as can be seen in Figure 1. The most external one is the *client layer* which is formed by the web browser tool through which the users (students and teachers) could access the system developed with this framework. The middle one, that is the *server layer*, corresponds to the part of the system deployed in a J2EE application server. This layer is in turn organized into three tiers. Lastly, the most internal part of the architecture, i.e. the *data layer*, refers to the information stored by the system and is composed of the student model repository and the knowledge base. The first one contains all the student models of the users of the system. The knowledge base has the domain information expressed by means of rules and problems.

The server layer is structured into the following parts: **a)**

The *presentation tier*: This part contains all the interface components. Here developers should construct a student module for presenting the problems and an authoring tool to elicit the domain. We suggest the use of *Java Server Faces* (JSF) technology for this purpose. JSF is a user interface component framework which facilitates the construction of sophisticated and interactive web applications. With this technology web-based applications look like java interfaces developed with Swing API. **b)** The *business logic tier*: This side is composed of all services which are in charge of: selecting the most suitable problem to be shown, inferring the constraints fired, assessing the student and determining whether the problem solving session should finish or not. **c)** The *persistence tier*: This part of the framework supplies access to the information stored in the data layer and is in charge of ensuring the consistence between information of the student model repository and the knowledge base, and the data managed by the rest of the services. This is done by a persistence management service, we have developed for this purpose. Accordingly this tier is the container of the student and the domain models, and has been implemented using the *Java Persistence API*.

3. Use Case: The Simplex tutor

Using the previous framework, we have implemented the Simplex tutor2. This tool allows teachers to assess their students while they are solving problems in which they have to apply either the Simplex or the Two-Phase algorithms. The Simplex method consists of an iterative algorithm, proposed by Dantzig in 1940, where the problem must be represented by means of a table which contains all data relating to the problem. Between two different iterations, the data in the table is changed in order to find a point (or points) whose value is maximum or minimum for a given function, known as an objective function. Different kinds of problems exist which can be classified by their difficulty depending on the skills needed to solve the problem [2].

The operating mode of this system is as follows: Once a student is registered, the system poses him/her a problem. This problem is generated randomly the same way as in [2], and must be solved by constructing a solution in different phases. We have separated the process into three main phases. The first one consists of transforming the problem into a standard form, i.e. rewriting the problem into another equivalent one with the required format according to the Simplex algorithm. Once the problem is in this format, it is put in a table in order to start the *Simplex* or the *Two Phase method*. This is a decision that must be taken by the student. The second step consists of iterating over the table obtained in the previous phase. The last phase occurs when the iterations have finished and the student must identify a solution.

Figure 2. Simplex Tutor Interface.

4. Conclusions

The main contribution of this paper is a framework to develop problem solving assessment tools.

We have tried to profit from the most recent technologies of distributed and loosely coupled architectures. In this sense, we have constructed this framework as a service oriented architecture. Therefore, we offer a set of services and accordingly developers should focus mainly on the presentation tier components and the domain elicitation.

For a first validation of this framework, we have developed a tutor which assesses the students while solving problems of linear programming using the Simplex or the Two Phase algorithms. We have used this framework because this is a domain where the final solution is not enough to determine the student's knowledge level, since it consists of a simple value, i.e. right or wrong. The most relevant information to evaluate the students can be found at the solution building process.

Within a short time, we plan to use this diagnosis tool with real individuals in a group of university students. Regarding future research lines, we are working on new components to automate the presentation layer construction and the elicitation of the domain model.

References

[1] T. Erl. *Service-Oriented Architecture : Concepts, Technology, and Design*. Prentice Hall PTR, 2005.

[2] E. Millán, E. García-Hervás, E. Guzmán, Ángel Rueda, and J.-L. P. de-la Cruz. Tapli: An adaptive web-based learning environment for linear programming. In *CAEPIA*, pages 676–685, 2003.

[3] J. Pellegrino, N. Chudowsky, and R. Glaser. Knowing what students know: The science and design of educational assessment. 2001.