Deliverable N°: D16.0

LeActiveMath
Integrated Adaptive Assessment Tool

The LeActiveMath Consortium

July 2005

Version

Main Authors:

Eduardo Guzmán, Enrique Machuca, Ricardo Conejo (Universidad de Málaga)
Paul Libbrecht (DFKI)

Project funded by the European Community under the Sixth Framework Programme for Research and Technological Development

Project ref.no.	IST-507826
Project title	LeActiveMath- Language-Enhanced, User Adaptive, Interactive eLearning for Mathematics

Deliverable status	Draft
Contractual date of delivery	
Actual date of delivery	07-21-2005
Deliverable title	Integrated Adaptive Assessment Tool
Type	Internal
Status & version	
Number of pages	46
WP contributing to the deliverable	WP3
WP/Task responsible	T3.7
Author(s)	Eduardo Guzmán, Enrique Machuca and Ricardo Conejo (Universidad de Málaga). Paul Libbrecht (DFKI)
EC Project Officer	Colin Stewart
Keywords	assessment tool, Siette

Contents

1 Executive Summary **5**

I Introduction to Siette System **6**

2 Theoretical Fundamentals **6**

3 Siette's Architecture **7**

4 Knowledge Base Structure **8**

4.1 Subjects and Topics 8

4.2 Items 9

4.2.1 Internal Items 9

4.2.2 Siettlets 10

4.2.3 Generative items 11

4.2.4 External items 12

4.3 Tests 13

4.3.1 Conventional test 14

4.3.2 Adaptive tests 14

5 Student Model Repository **15**

6 Functionalities **16**

6.1 Content creation 16

6.2 Student registration 17

6.3 Test administration 17

6.4 Result analysis 18

6.5 Item calibration 19

7 Implementation **20**

II Integrating Siette into LeActiveMath **21**

8 Uses of Siette in LeActiveMath **21**

9 Content Exchange **22**

9.1 Siette's Questions and Test Interoperability File 22

9.2 Automatic Generation of S-QTI files 25

10 Loose Coupling **26**

10.1 Loose Coupling Requirements 26

10.2 The Browser Delegation Scenario 27

11 Web Service-based Actions **29**

11.1 Services offered by Siette 29

11.2 Events Issued by Siette 30

11.3 A Generic Case of Use 31

12 Future Work **32**

III Description of the S-QTI Schema File **33**

13 `type_subject` type **33**

14 `list_topics` type **33**

15 `type_topic` type **33**

16 `list_translations` type **34**

17 `type_translation` type **35**

18 `list_items` type **36**

19 `type_item` type **36**

20 `type_iccparameters` type **38**

21 `list_responses` type **38**

22 `type_response` type **39**

23 `type_blank_response` type **39**

24 `type_evaluation` type **40**

25 `type_eval_response` type **40**

26 `list_tests` type **41**

27 `type_test` type **41**

28 `list_permissions` type **43**

29 `permissions` type **43**

1 Executive Summary

Siette is an adaptive test-based assessment tool. One of its most relevant features is that it administers pre tests which would provide student models with initial values. However, Siette can be also used as an assessment tool during the formative process (i.e., during the student instruction), as an external resource for self-assessment, or for summative evaluations [20] about the student's achievements in the different topics of a curriculum.

Siette has been integrated into LeActiveMath to provide well-founded assessments, based on adaptive test administration. When administering adaptive tests for LeActiveMath, Siette is in charge of selecting the most adequate question which must be posed to the student, to infer his/her knowledge level, and to determine when the test must finished. Question presentation is a made by LeActiveMath itself on Siette's demand.

The tasks accomplished to make this integration were described in [17], and are briefly summarized below:

- *Adaptation between* LeActiveMath *and* Siette *knowledge representation.* This task has been carried out by developing an automatic translation process. This translator generates a XML-based file from LeActiveMath's knowledge base in a specific format understandable by Siette, called S-QTI.
- *Design and development of communication protocols and interfaces.* A loosely coupling integration mechanism has been developed to this end.
- *Integration into* LeActiveMath . The former integration mechanism allows a transparent integration of Siette's assessment tests into LeActiveMath courses. Consequently, students do not notice they are using different systems during the instruction process.

This document is structured into three main parts. The first one is a description of the Siette system. The second part explains how the integration has been carried out. Finally, the third part depicts the structure of S-QTI schema file. In this project this schema is used to interchange knowledge content between LeActiveMath and Siette.

Part I

Introduction to Siette System

Siette stands for *System of Intelligent Evaluation using Tests for Teleeducation* in Spanish [6, 14]. Siette is a test-based assessment system that has been designed to be used through the WWW.

Two different user classes can take advantage of Siette. On the one hand, teachers can define subjects, their topics and items, and specify tests, making them available to students. They can also use Siette to make academic assessments. In a controlled environment (such as a laboratory with PCs connected to the Internet), students can be assessed with Siette. The use of Siette has some advantages: tests tailored to students' personal abilities, automatic correction of tests, on-line grading generation, few software requirements (just a web browser tool), etc. To prevent cheating and unauthorized accesses, Siette incorporates several security mechanisms such as test access restrictions by groups, IP addresses or users. Finally, they can analyze the performance of the students that have taken tests. On the other hand, students may use Siette as a way of self-assessing their ability in the subjects taught by teachers.

Features of Siette such as reliability and scalability have been tested out with hundreds of real students in the School of Computer Science, the School of Telecommunications in the University of Málaga (Spain) and in the Technical School of Botanics in the Polytechnic University of Madrid (Spain). In these Schools SIETTE is often used by lecturers for academic grading [13].

Siette is an adaptive test delivery tool. In adaptive tests, each student is administered a different test. Items are dynamically selected and presented to the student in terms of his/her knowledge level estimation. The goal is to obtain accurate student knowledge estimations and to minimize the number of items required to this end. For this reason, after each student's response, the estimations are checked to see whether they are accurate enough to stop the test.

Siette can also operate as a conventional assessment tool, i.e. using fixed tests where the number of items posed to students are the same for all students. These tests are assessed with criteria like the percentage of correct items over the total number of items, or with scored items, which are the classical heuristics used by other tools.

2 Theoretical Fundamentals

Any instructional process should be complemented with assessments of the degree of assimilation of the topics (or concepts) studied. In intelligent tutoring systems, assessment is even more relevant, since these systems require some knowledge information sources to guide the instruction. Ideally, each exam should be adapted to the personal circumstances of each individual student. On the whole, from a practical point of view, adaptation can only be carried out in environments in which the number of students is very small. Adaptive tests (a.k.a. *Computerized Adaptive Tests*) represent an attempt to automate this difficult task, since with them students can be assessed independently. The main idea of an adaptive test is to act the same way a teacher would [21] when he/she assesses orally. If a teacher asks a student a question (so-called item) that turns out to be too difficult for him/her, the next question to be posed must be easier and vice versa.

Item Response Theory (IRT) has become the main basis of this measurement theory. IRT [16] rests on two principles [7]: the performance of a student in a test can be explained by a set of factors called latent traits, which can be measured by means of unknown fixed numerical values; and the relationship between student item performance and the set of underlying item performances can be described by functions called *Characteristic Curves* (CCs). The CCs represent the conditional probabilities of the successful answer to the item (or the selection of a certain answer) by a student

with a certain latent trait (θ) measured in the domain of real numbers. CCs must be previously known for each item, and are expressed by means of a probabilistic function. In the field of adaptive testing, the latent trait is the knowledge level. There are several IRT model classification criteria, e.g., in accordance with the number of latent abilities simultaneously measured, depending on the shape of the CCs, etc. As a consequence, there are many IRT models. One of the most commonly used models is the three parameters logistic model (3PL) [8]. The CC in the 3PL is modeled according to the following equation:

$$P(u_i = 1|\theta) = c_i + \frac{1}{1 + e^{-1,7a_i(\theta - b_i)}} \quad (1)$$

$u_i = 1$ indicates that the student has answered item i correctly. In another case ($u_i = 0$), the probability is equal to $1 - P_i(\theta)$. This model receives its name because it is characterized by the three parameters below:

- *Discrimination factor (a_i)*: A high value indicates that the probability of success by students with greater knowledge than the item's difficulty is higher.
- *Difficulty (b_i)*: This parameter corresponds to the knowledge level in which the probability of answering correctly is the same as answering incorrectly.
- *Guessing factor (c_i)*: This probability suggests that a student without any knowledge will answer the item correctly and represents the case in which a student answers randomly.

In Siette the knowledge level is measured using a discrete IRT model. Instead of taking real values, the knowledge level takes K values (or latent classes) from 0 to $K - 1$. Teachers decide the value of K in terms of the assessment granularity desired. Likewise, CCs are turned into probability vectors $p(u_i = 1|\theta = 0)$, $p(u_i = 1|\theta = 1)$, $p(u_i = 1|\theta = 2)$, ..., $p(u_i = 1|\theta = K - 1)$.

IRT has been successfully applied to the item selection mechanisms and student's knowledge level estimation in adaptive testing. The results obtained are independent of the tool used. This measurement is invariant with regard to the sort of test and to the individual who takes the test. The main drawback of IRT is that the parameters of the CC must be previously known for each item. These parameters are obtained initially from estimation techniques. The problem is that these techniques use the performances of students that have taken a test with these items non-adaptively.

3 Siette's Architecture

The architecture of Siette [11], comprises the main components that usually appear in any cognitive diagnosis tool system. The following parts can be mentioned:

- *Knowledge base.* Its contents are organized into subjects (or courses). A subject is broken down hierarchically (in a tree) into topics (or concepts), forming a curriculum. Items can be defined associated to the topics. Siette can include different types of items [12]: true/false, multiple choice, multiple response, open-answer, etc. Test specifications are defined in accordance with the topics they assess.
- *The student model repository.* This is a collection of student models. Each student model stores the information about a student's test session (knowledge probability distributions, knowledge level estimation, item posed, exposure time per item, etc.). The test generator dynamically updates these data after each response. The item calibration process can be carried out thanks to the information stored in the student models.

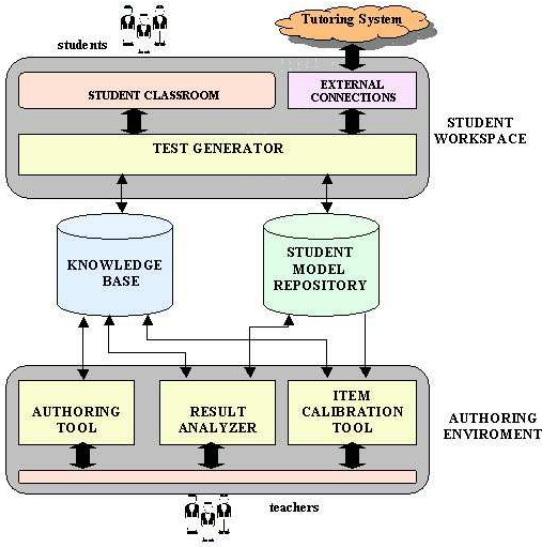

Figure 1: The architecture of Siette.

- *Student classroom.* This is the virtual environment where students take tests.
- *Test generator.* This component dynamically constructs test sessions. A test session is a test tailor-made for a student according to the specifications stored in the knowledge base.
- *External connections interface.* Siette not only works as an independent assessment tool, but can also be integrated into other web-based tutoring architectures as an additional diagnosis module, providing well-founded assessments.
- *The authoring tool.* This is a Web-based utility to add to and update the contents of the knowledge base [15].
- *Result analyzer.* This utility allows teachers to analyze the performance of students in tests. Moreover, this tool provides a service for consulting the student models.
- *Item calibration tool.* This module uses the information obtained from the student model repository to calculate the psychometric properties of the items. The inference process of these values is essential for administering of adaptive tests.

4 Knowledge Base Structure

The knowledge base is the expert module of Siette. It is mainly composed of the following four components: subjects, topics, items and tests.

4.1 Subjects and Topics

In the knowledge base, domain models are structured into *subjects*. They represent the highest level of abstraction in content definition. In turn, *subjects* may be divided into different *topics*. A *topic* can be defined as a concept about whose knowledge students can be assessed. They can also be broken down into other topics and so on, forming a hierarchy where the degree of granularity is decided by the teacher. In this hierarchy, leaf nodes represent a unique concept or a set of

concepts which are inseparable from the assessment point of view. Topics and their subtopics are related by means of aggregation relations (i.e. *part-of relations*), and no precedence relations are considered. Figure 2 depicts an example of a domain model, an item pool, a set of tests and the relations among them.

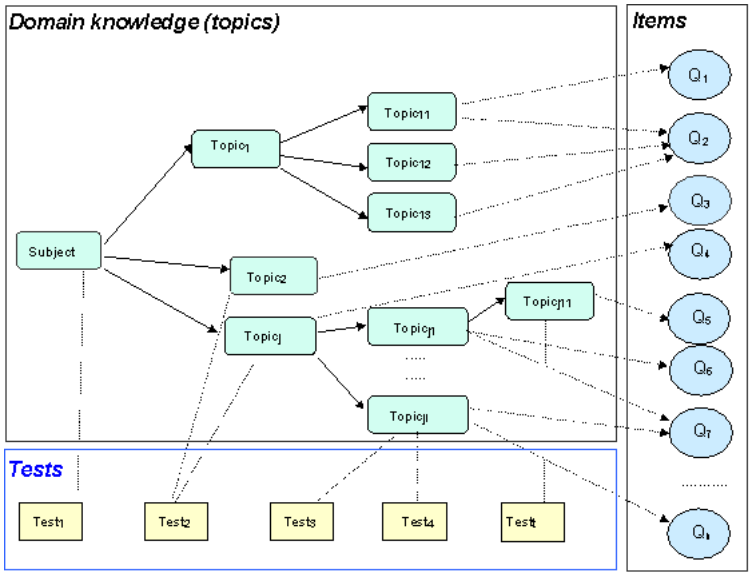

Figure 2: Subject curriculum, items and tests in Siette.

4.2 Items

Items are not only the classical test questions, but also exercises. In Siette, an item is associated to one or more topics. This association expresses how this item provides information about the student knowledge level in these topics, as a consequence, items can be considered evidence providers. The collection of items which assess a topic is called an *item pool*. Figure 2 shows on the right side the whole item pool of a subject. Teachers are allowed to administer different types of items, all of them into the same test session. There are several ways of categorizing items, but none of them can be completely fitted into items available in Siette. In this report, the taxonomy proposed in [19] will be followed. All of the items of Siette belong to one of the internal types. In addition, items can be (or not) siettlets, generative and/or external.

4.2.1 Internal Items

- *True/false items*: These items ask about the certainty of a statement. It can only have two answers: true (correct) or false (incorrect).
- *Multiple-choice items*: This kind of item presents more than two possible answers (choices). Examinees may select one (or none) of these answers.
- *Multiple response items*: These items are multiple-choice items with more than one correct choice. Students must select all correct choices in order to pass the question. This kind of item can be further classified into:

  a) *Items with independent choices*. Choices are mutually independent. This type of item is equivalent to a set of true/false items.

b) Items with dependent choices: The correct answers are combinations of the set of possible answers. For the answer to be correct, all members of a combination must be selected. This kind of item is equivalent to a multiple-choice item where the answers are all possible combinations. That is, if the item has n possible answers, it is equivalent to a multiple-choice item with a factorial of n possible answers.

- *Open answer items*: In this type of item, students must write the answer (or answers) that satisfies certain conditions. In this case, answers stored in the knowledge base are response patterns, either to identify correct answers, or incorrect answers. In addition, a default pattern is always included, that represents an incorrect response which does not satisfy any other pattern. An evaluation procedure is applied, in terms of the open answer type, for each one of the answers written by the student.

  Depending on the way in which the written answer is corrected, there are three types of open answer items:

  – *Matching-based:* Answers supplied by the student are directly compared to the response patterns.
  – *Regular expression-based:* During the item construction process, patterns stored are regular expressions. The item correction procedure consists in checking, for each response pattern, whether the set of student's answers belongs to the language generated from the corresponding regular expression.
  – *Java regular expression-based:* These are analogous to the former items, but allow the construction of more complex patterns. These patterns are generated according the Java language regular expression class. This class is available from the 1.4 version of Java language.

  Three former types of open answer items have been constructed as plugins. Consequently, it is easy to include new types. In order to do this, it is enough to extend an abstract class with the correction mechanism desired. This abstract class only requires the implementation of a method which determines whether a certain string belongs to a certain pattern or not.

  This type of item allows the system to store incorrect response patterns. These patterns are particularly useful in self-assessment tests, since they bring about the opportunity to identify misconceptions and generate adaptive feedbacks. Incorrect patterns can be inferred applying bootstrapping during the item calibration process. These patterns can be obtained by analyzing the performances of students that had previously taken tests with the item.

  In tests where the item correction is shown immediately after the student's response, for those cases in which the student answers incorrectly, it is necessary to include an example of a correct response. This must be added during the item construction stage.

  From the assessment perspective, this type of item can be considered as a collection of multiple choice items with a size equal to the number of boxes shown within the item.

4.2.2 Siettlets

Items' interfaces are pieces of HTML code in Siette. Therefore small programs embedded in Web pages (like Java applets or Flash movies) can be added to the stem or to any of the choices to enhance presentation. Siette provides another kind of item, called *siettlets* (formerly known as self-corrected items), where the assessment mechanism is accomplished by an embedded program. This type of question does not offer a list of possible responses. In this case, the student must interact with the embedded program. Student actions are captured and processed by the program to determine whether the answer is correct or not. Several specific tests have been developed in Siette using this kind of item, like a Piagetian test for cognitive ability estimation [3], and a test of European trees and their geographical distribution.

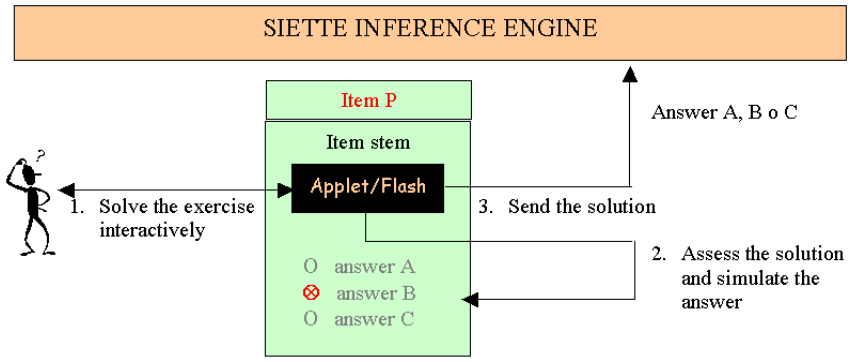

Figure 3: Assessment process by a siettlet.

Figure 3 shows a schema of the assessment mechanism in this kind of item. Firstly, the item is shown to the student. He/she interacts with the program inside the item. Once the student has solved the problem, the applet corrects his performance, and then it internally selects the corresponding (hidden) choice. Subsequently, the accuracy data is automatically sent to the assessment mechanism of Siette. See [3] for a complete description of this mechanism. These kinds of items can also be classified as an internal item type, depending on the number of possible responses that the embedded program uses to assess the student's answer. Thanks to this type of item, Siette offers the possibility of including virtually any kind of item (as long as it can be implemented by means of a Web embedded program), keeping the assessment mechanism unchanged. Certainly, this possibility is restricted to testing developers with some programming skills.

4.2.3 Generative items

To have a valid test-based delivering system, the item pool should contain a very large number of items (at least 500 are recommended) [9]. Excessive item exposure might cause students to learn the correct answers and even share them with the remaining students. This must be avoided to guarantee a valid assessment. Therefore, it means that teachers must add a huge set of items to the pool. This is a very hard and time-consuming task. A partial solution to this problem is the automatic generation of items. A review of the generation problem can be seen in [4].

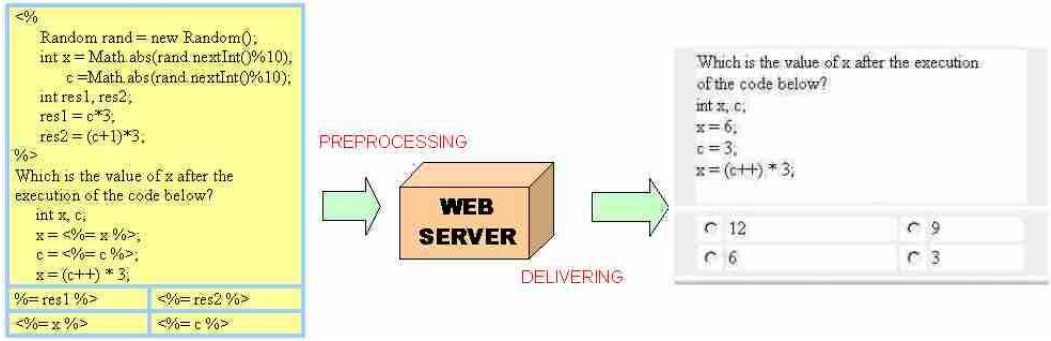

Figure 4: Item generation and delivering from an item template.

Siette makes it possible to create templates to generate isomorphic items in real time. When the teacher develops a template, a different item is generated each time it is administered to

an examinee. The templates can be implemented by using HTML-embedded languages, such as PERL, PHP, JSP, and so on. The inclusion of a generative item is done transparently. In the creation of an item, the teacher must indicate whether the item is generative or not, by marking an option in the authoring tool. Instead of adding HTML code in the stem and the answers, the teacher must add pieces of the HTML-embedded languages with random value generation sentences. These sentences are in charge of generating the correct and incorrect answers according to the stem. Before the item is shown, a pre-processing step is carried out. A buffered web page is composed of the stem and the answers. This page is sent to the web server with the appropriated plug-in to support the embedded language. The web server generates a page with just HTML code. This page is returned to Siette instead of being shown. The stem and answers are separated again, and composed in the final web page that will ultimately be displayed to the examinee. An example of a template in JSP is shown in Figure 4. It is a question about the Java language. The main code of the template, as well as the statements, are stored in the stem. The code that generates each answer is stored in the place of the answers. The values of variables x and c are generated every time the item is administered. Note that this mechanism allows generating distractors, i.e., answers that even though they are not correct, are similar to the correct answer. In the item generated in the right side of Figure 4, the correct response is 9, but if the student has insufficient understanding of Java, there is a higher probability of his/her selecting an incorrect answer. Some IRT-based models are able to take into account the response selected by students and make an estimation according to the degree of error. Currently Siette does not support these models. The generation mechanism has the same problem as the self-assessment items; namely, that teachers must have some programming skills.

4.2.4 External items

By means of this type of item, Siette allows the inclusion of items that are actually stored in external Web-based systems. They are similar to siettlets in the sense that they correct the student's response and send this response back to Siette. In spite of the content of these items which are not stored in the knowledge base of Siette, there is some information that must be stored. For each external item, the URL through which the item will be called and the set of parameters required for this call, are collected. Furthermore, the set of possible responses given back from the external system (item responses) are also stored.

When during a test, one external item is selected, in order for it to be posed to a student, Siette will call the corresponding URL, passing on the set of parameters needed. Consequently, Siette temporarily loses the execution control, giving this to the corresponding external system. Once the student has answered the item, the external system must invoke a specific URL of Siette formerly passed on as a parameter.

Figure 5 represents the functioning mechanism of an external item. The left side of the figure shows how the Siette system includes M external items. The right side shows (in different colors) the external items in their original locations. The figure depicts how Siette only stores the external item URL location and the set of responses given back from the external system. The set of intermediate arrows represents the call flow between Siette and the external systems during a test. As can be seen, Siette invokes the corresponding URL direction, passing the callback URL through the "url" parameter. When this callback happens, the external item must provide Siette (through the "answer" parameter) with the the name of the parameter which has sent the response.

External items can be combined in a test with any other item type. ¿From the assessment point of view, these kinds of items will belong to one internal item. The use of internal items requires, from the external system side, the capability of giving the answer back to the URL passed on as a parameter.

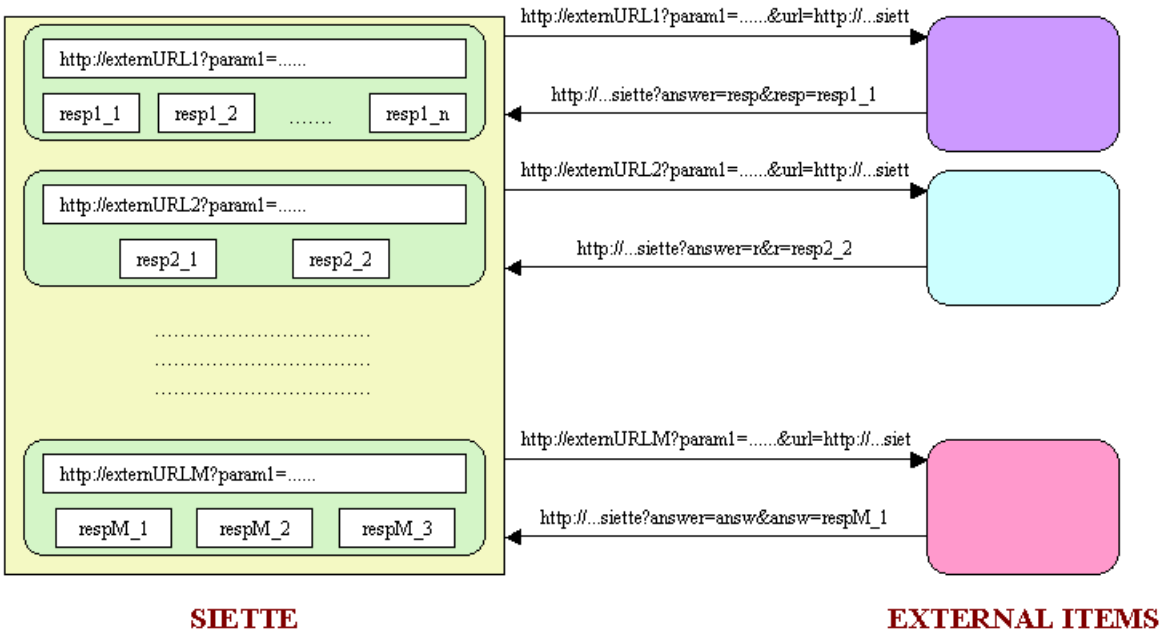

Figure 5: External item functioning diagram.

4.3 Tests

In this system, the student knowledge diagnosis is made using tests. A test is a specification of an assessment session, based on a set of configuration parameters. From these parameters, the test generator will create assessment sessions on demand. The key goal is to infer the student's knowledge level in one or more topics of the curriculum. The lower of Figure 2 shows the test specification collection for a subject.

When creating a test, a teacher must answer several important questions, which summarize its goals and features [5]. These questions are:

1. *What assess?*, i.e., which topics of the curriculum are going to be assessed.
2. *Who assess?*, i.e., what particular characteristics the student has who is to be assessed. This information will be stored in his/her student model.
3. *How assess?*, that is: *a)* Which assessment criterion is going to be used. *b)* The grading scale used, that is, the number of the knowledge level in which students will be classified. And, finally, *c)* How will the items be sequenced?, i.e., the item selection criterion used in the test.
4. *When must the assessment process finish?*

All these considerations are materialized in test parameters. Tests are defined mainly in terms of the topics they assess.

As mentioned earlier, Siette offers two kinds of tests: conventional and adaptive tests. The main difference between them rests on their validity. Adaptive tests are based on well-founded theories. These theories ensure students' knowledge inference will be kept unchanged if another adaptive test (about the same topics) is applied, and if students are not involved in any instructional process between test administrations. In contrast, conventional tests are easier to apply, but lack a theoretical base, i.e. they are based on heuristics.

4.3.1 Conventional test

These are the most commonly known tests. Their construction is easier than the adaptive test construction, since the requirements needed a priori are significantly lesser. In Siette, the following assessment criteria have been defined for these tests:

- *Percentage-based criterion*: It consists in calculating the percentage of items answered successfully.
- *Score-base criterion*: In this criterion each item has a score assigned which is given by the teacher during the creation stage. This score can be assigned either to the global item or to each one of the possible choices. The student's final qualification is expressed by means of a percentage regarding the maximum obtainable score.

Once the percentage is calculated by one of the former criteria, this value is mapped onto the test knowledge level scale.

Regarding the topic and item selection, in this kind of test, item selection is made heuristically, according to a percentage predefined for each topic by the teacher in the test creation stage. This set of percentage tries to ensure that the number of items in an assessment session will be balanced. Once the topic has been selected, the item that is going to be posed is chosen according to one of the following item selection criteria:

- *Random*: Items are selected randomly.
- *Difficulty ordered*: Before the first selection, it orders all items in the pool in ascending order of difficulty. The least difficult available item is selected each time.
- Ordered: Sometimes teachers prefer that the selection process follow a pre-established order, which they will have previously set out.

Finally, in this kind of test, the following finalization criteria can be used:

- *Maximum item number*: This criterion determines that the test must finish once the student has been posed a certain number of items. This threshold is configured in the test.
- Time limit: Students have a maximum global time to answer to all test items. Once this time expires, the finalization is forced.

4.3.2 Adaptive tests

In Siette, adaptive tests are based on IRT. In this kind of tests student knowledge estimations are represented by means of probability vectors, where each vector is a table that collects pair knowledge level-probability. As will be seen in the next section, these probability vectors are stored in student models. Initially, at the beginning of the test, if the student has not made any previous tests, Siette assumes that all the vector knowledge levels have the same probability value. Once the student has answered an item, the CC corresponding to the student's response is used to update his/her corresponding knowledge level:

$$P(\theta|u_i) = P(\theta)P_{u_i|\theta}(\theta) \tag{2}$$

¿From the knowledge probability distribution, the student's knowledge level can be inferred in two ways, depending the criteria used to this end:

- *Maximum A Posteriori* (MAP): The knowledge level is that with the most probability value.
- *Expectation A Posteriori* (EAP): The knowledge level is the expectation of the probability distribution.

In this kind of test topic and item selection is done adaptively and simultaneously. The goal of both selection processes is to minimize the number of items required in the tests and, at the same time, improve the estimation of the student's knowledge level. In Siette, the following IRT-based topic and item selection criteria have been defined:

- *Bayesian*: This criterion selects the item that minimizes the expectation of variance of the a posteriori student's knowledge distribution. [10] reveals that this criterion is not only able to select the most adequate item, but it is able to select the most adequate topic.
- *Difficulty-based*: This method is very similar to the former, but computationally more efficient. It first selects the topic whose student's knowledge distribution variance is higher, and from this topic, the item whose difficulty is closer to the student's knowledge level in the corresponding topic.
- *Entropy-based*: This criterion is based on the Shannon's concept of entropy. Consequently, it selects the item that minimizes the a posteriori entropy of the student's knowledge distribution.
- *Maximum information function-based*: This method is analogous to the difficulty-based one. It first selects the topics whose student's knowledge distribution variance is higher, and from this topic, the item whose information function in the student's knowledge level is the highest.

Finally, in adaptive tests, finalization criteria can also be applied adaptively. However, adaptive finalization criteria should be applied together with any of the criteria described in the former section. The adaptive criteria included in Siette are the following:

- *Maximum reachable precision*: This method consists in setting a threshold that will be used to determine the maximum allowable variance of the student's knowledge distributions. Once this threshold has been exceeded the test will finish.
- *Minimum probability*: In this method a threshold is also set. This threshold is compared to the probability associated to the student's knowledge level. If the threshold is lesser, the test will finish.

5 Student Model Repository

This repository stores the models of the students that have taken tests through Siette. Test generator updates these tests during their administration. Consequently, teachers can consult these models through the analysis tool. The item calibration process is carried out using the information stored in this repository.

As can be seen, each model stores vectors with the knowledge probability distributions in each one of the topics assessed in the tests, that he/she has taken. For each student, the repository stores the set of items administered per test, the order in which the items where posed, and the response selected by the student. Additionally, other information is stored: the IP direction from where the student took the test, the starting date, the time expended on each item, etc. All this information is kept permanently. Only authorized teachers can delete information about the student that took some of their tests. This functionality is only provided by means of the analysis tool.

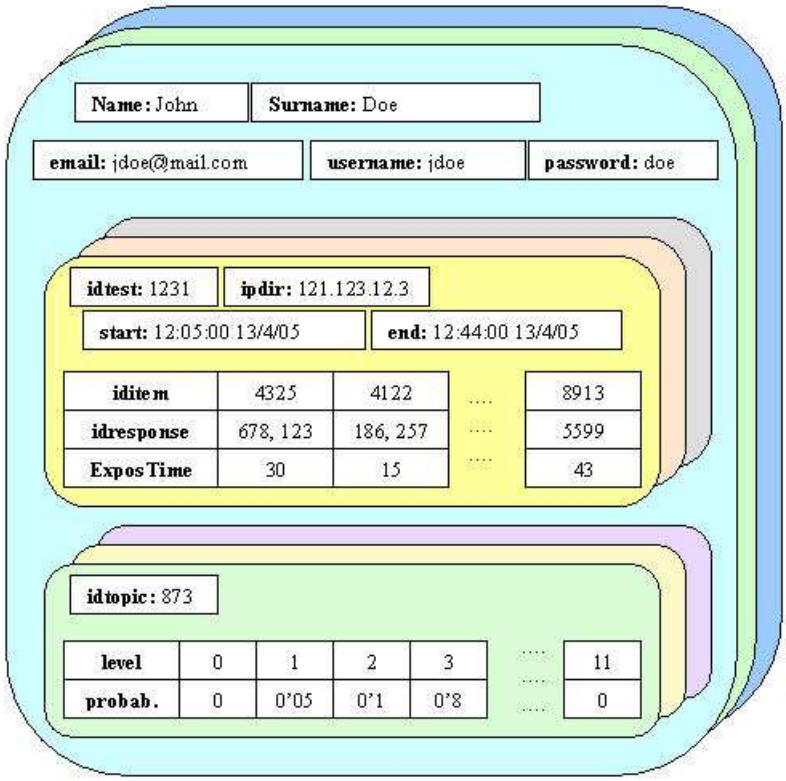

Figure 6: Student model repository in Siette.

6 Functionalities

Siette provides a wide set of functionalities related to the content (subjects, topics, items and/or tests) elicitation and update, test administering, student management, etc. This section emphasizes just those functionalities closer to LeActiveMath requirements.

6.1 Content creation

This functionality can be accomplished in two different ways. As described earlier, Siette includes a Web-based authoring tool where teachers can create and update subjects, topics, items and tests.

Furthermore, Siette allows teachers automatic content addition to the knowledge base through XML-based interchange files. Each one of these files contains the whole specification of a subject including their topics, items and tests. An international organization, the IMS Global Consortium, exists. One of its goals is to standardize learning technologies. They have some specifications considered de facto standards. One of these specifications is about tests and questions: *Question and Test Interoperatibility* (QTI). From the Siette's point of view, this specification is not sufficient to express all features provided by the components of the Siette's knowledge base. For this reason, Siette supplies its own specification, called Siette-QTI (S-QTI). To generate XML files in accordance with S-QTI, there is an XML-Schema that allows file validation. This schema is described in detail in the last part of this report. Furthermore, Siette provides with a mechanism to generate the S-QTI file of a whole subject, and the opposite procedure, i.e., the content of a S-QTI file can be updated in the knowledge base.

Finally, Siette gives an off-line converter tool prototype that generates S-QTI files from IMS's QTI files and vice versa. However, there can be loss of information when a S-QTI file is converted into an IMS's QTI file.

6.2 Student registration

In Siette, students must be registered before taking any test. Registration can be done by any user, through the virtual classroom website. To be registered, Siette only requires a pair username/password suggested by students. Additionally, students can provide his/her first name, family name and email.

For most of the tests, student registration is enough to gain access. Furthermore, Siette supplies a mechanism to restrict tests to one or more groups of students. Consequently, this kind of test can only be accessed by students included in these groups. Teachers are responsible for group management. This can be done by means of the authoring tool.

6.3 Test administration

A *test session* is the administration of a test to a student. All test sessions share the features and properties of the test, but each session can differ from each other in terms of the number of items posed, etc.

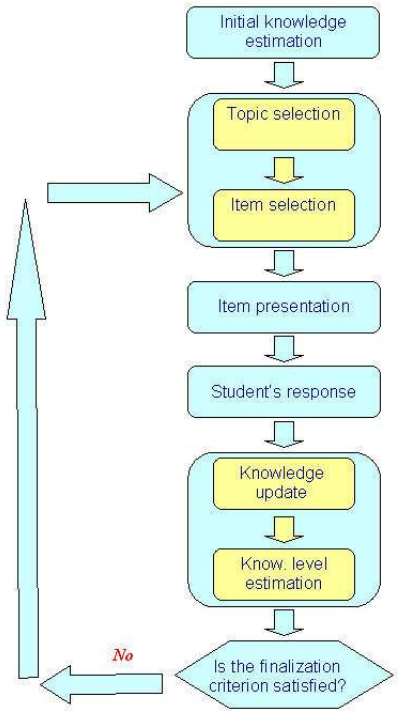

Figure 7: Test administration flow diagram.

The flow diagram of a test administration is depicted in Figure 7 and consists of the following stages:

1. *Initial estimation of student knowledge*: In this step the student model is initialized.

2. *Topic selection*: The topic on which student's knowledge will be assessed is selected.

3. *Item selection*: Once the topic to be assessed is chosen, this step selects the item (not yet administered) that will be posed to the student.

4. *Student's response*: The item is asked and the student responds. Figure 8 shows what an item looks like, in the virtual classroom. The left window shows how the item is posed to the student, and the right window shows the item correction such as it is revealed to the student. Green ticked choices are those which were selected by the student and are correct. Red ticked are the correct answers not selected by the student. Finally, red crosses indicate the choices selected by the student that are incorrect.

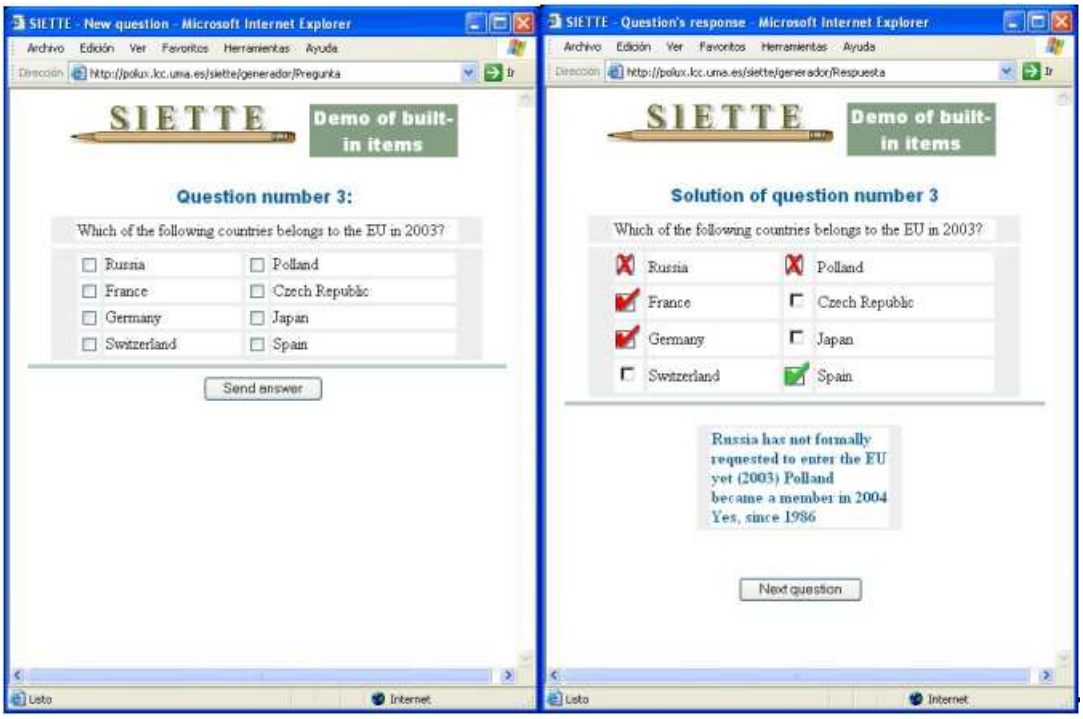

Figure 8: The virtual classroom.

5. *Student knowledge update*: In accordance with the student's response his student model is updated.

6. *Student knowledge level inference*: The student's knowledge level is inferred in the topic assessed.

7. *Evaluation of finalization criteria*: A check is done to see whether or not the finalization criterion (or criteria) is satisfied. If this is not the case, the process will continue from step 2, until the finalization criterion is met.

6.4 Result analysis

This functionality allows teachers to inspect student models. To this end, Siette supplies a Web-based result analyzer tool. Using this tool, teachers can consult the students' performance in tests and generate statistics about the results

A student performance facility For each test it contains the list of students that have taken the test. For each student it shows the identifier of the test session, his/her identifier and name, the date of the beginning of the test session, the total number of items posed, the number of items correctly answered and his/her final grade. It allows the viewing of the complete test session, that is, the items that were given to the student in the same order posed, with the student's response and the correct response. This tool gives detailed statistics of the final student knowledge level estimation. For each topic, the estimation, the number of items posed and the number of correctly answered topic items, as well as a graphical representation of the estimated knowledge distribution, is provided. Additionally, it offers the possibility of deleting student test information from the student model repository. Figure 9 shows several snapshots of this tool.

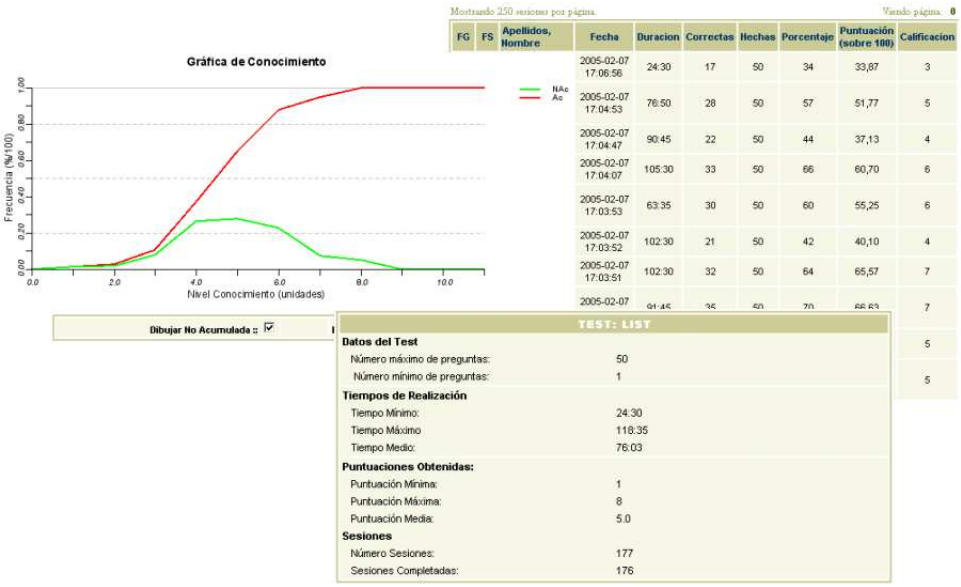

Figure 9: The result analyzer.

An item statistics facility It supplies statistical information about student responses to the item in all test sessions in which the item has been posed. These data can be studied for each topic to which the item is directly associated, and for each one of its preceding topics, including the subject. Once the topic to be studied has been selected, a table is shown. It contains a column for each item answer. Each row represents a knowledge level in which the subject can be assessed. Each cell c_{ij} of the table represents the number of students with final estimated knowledge level i that have selected the answer j. In addition, cumulative statistics are shown: that is, taking all the data of the student model repository as a sample, the likelihood that a student will select an answer given his/her final estimated knowledge level. This information is very useful for the calibration purposes.

6.5 Item calibration

As mentioned before, items must be calibrated in order to be used in adaptive tests. Calibration processes are carried out from the results obtained by students that have taken conventional tests with the items to be calibrated. For this reason, Siette includes a calibration tool, although it is currently under development. The objective is to provide a Web-based calibration tool. Now calibrations are made using an off-line tool. For more information about the algorithm used by Siette to accomplish the calibration see [14].

7 Implementation

Siette is a Web-based tool developed in Java following the J2EE (*Java 2 Enterprise Edition*) specification. This specification has been developed by SUN Microsystems, to construct distributed Web-based applications. Accordingly, Siette is implemented using combinations of JSP and servlets, with a kernel of classes that comprises their middleware. The programming model used in Siette is the *Model-View-Controller* paradigm. This paradigm is often used in the development of software following the J2EE specification.

Currently, Siette is installed in a Web-server for a J2EE application, *Apache Tomcat*. Both the knowledge base and the student model repository are implemented on a relational database, using the database management system developed by Oracle.

Part II

Integrating Siette into LeActiveMath

When two Web-based applications are complex, such as LeActiveMath and Siette, the need for a loose coupling arises. In these cases, actions such as moving the student from one application to the other, or the existence of two different student and domain models makes it difficult to accomplish tasks jointly.

Integration in most Web-based applications simply consists in including relatively transparent links between the systems, but when intelligent systems are integrated (such as LeActiveMath and Siette), the objective is not only the cooperation in presentation, interface homogeneity or single sign-on. Problems such as getting information back from one system to the other need to be solved. For instance, if Siette were to be integrated in LeActiveMath by simply including some links, Siette would not be able to return assessment results to LeActiveMath, and accordingly this information could not be used to update the LeActiveMath's student model. Another important issue is the fact that tests done in Siette must use the LeActiveMath's exercises, i.e. exercises are not located in the same place as Siette. Therefore, if coupling is not done properly, students may get stuck on an item and the test will not be completed, leading students to be in unacceptable situations.

An even more complicated issue in integration is the need for an automatic content exchange. Both LeActiveMath and Siette have their own domain model. Siette's domain model elements are references to LeActiveMath's domain model elements. For this reason, Siette must have updated this information. As a consequence, a mechanism is needed to notify Siette of the changes effected on LeActiveMath's domain model elements, i.e., exercises, topics and tests.

Consequently, a coupling is needed between both Web-applications. This coupling must be done in such a way that students should not notice that they are browsing in different systems. To achieve these goals, developers of both web applications need to specify communication protocols and information exchange scenarios.

This part of the document is structured as follows: the next section is devoted to describing how LeActiveMath can take advantage of Siette, i.e. in which learning stages the use of Siette can be valuable. After that, the problem of knowledge domain content consistency between both systems is approached, and also how this problem has been solved. Loose coupling will be described as a generic mechanism to integrate components in LeActiveMath's architecture. In this framework, services provided with Siette using this mechanism are detailed.

8 Uses of Siette in LeActiveMath

Maintaining a student model with values that approximate the estimated mastery of the student is a delicate task which relies, in LeActiveMath, on tracking the student's reading actions and receiving exercise diagnoses. Both of these methods, however, can only be achieved after some time using the learning environment whereas the ideal is to offer guidance as early as possible.

Originally, LeActiveMath initialized the learner model with default values (zero), and offered the student a self-assessment that provided other mastery values.

A better alternative is to invite the student for a test-based assessment session (i.e. a pre-test) in order to infer his/her knowledge on the topics involved in the course he/she wants to do. This can be done using the Siette system.

In addition to the former scenario, Siette can be used to provide formative evaluation. Consequently, during the instruction, contents may show links inviting the student to take a self-assessment test. After this test, LeActiveMath's student model should be updated regarding the student's test performance. This is the main scenario where integration takes place.

The decision on including links to tests, shall be taken by the LeActiveMath's tutorial component and the student model. The tutorial component is responsible for the content selection and further reading advice. During the interactions with the tutorial component, links inviting the student for an assessment session will be presented. When a link is clicked on, it will take the browser to an assessment session in Siette which upon completion returns the student's browser to the component where it was before.

After such a session, the student model should be updated as a result of the tests. Using this updated information, the tutorial component will be able to provide better advice on further material to be read or exercises to be carried out. After the assessment, the open student model will be able to present estimates more accurately, which should enhance the students' trust in the system capabilities.

9 Content Exchange

One requirement for the link inviting the session to be displayed is the recognition of the assessment tests available. To this end, the same topic IDs and domain model structure must be shared between both systems. Therefore, through a simple web service call, to retrieve a whole set of knowledge contents and exercises from LeActiveMath and to update Siette's knowledge base.

An export procedure, implemented in `SietteAsignaturaProducer` class (located at `org.activemath.util` package), has been developed in order to exchange contents between both systems. This procedure generates an S-QTI file understandable by Siette.

Furthermore, exercises do not reside in Siette any longer, but they rather are stored in LeActiveMath. Now, when Siette is requested for a test administration, it delegates the exercise presentation back to LeActiveMath. Once the exercise is presented, the student's evaluation is again delegated to Siette. This can be done using the Siette's external items.

9.1 Siette's Questions and Test Interoperability File

As mentioned before, there are two methods of content creation in Siette. On the one hand, by using the Web-based authoring tool; and on the other hand, by writing an XML file based on *S-QTI*. In this section, an example of an S-QTI-based file is described. This example corresponds to a real LeActiveMath book.

Subject information

```
<subject
xsi:noNamespaceSchemaLocation="http://www.lcc.uma.es/siette/xml/siette_english.xsd"
         id="-1">
  <name>mbase://LeAM_calculus/derivation_grouping</name>
  <numknowlevels>12</numknowlevels>
  <isactive>true</isactive>
  <topics>
  ...
  </topics>
```

```
  <items>
  ...
  </items>
  <permissions>
  <author>19</author>
  </permissions>
  <topicid>-261</topicid>
</subject>
```

Item information Siette's items used in LeActiveMath are external. The item shown in the next example, includes in its stem the URL needed to invoke its external presentation. To this end, the MBase (i.e. the knowledge base of LeActiveMath) identifier of the exercise must be known, as well as the responses patterns which will be used to evaluate the student's answer in Siette. LeActiveMath returns to Siette a number ranging from 0 to 100. From this response, Siette evaluates it by comparing this received value with the patterns (stored within the item information in Siette) to make an IRT-based assessment of the student.

```
<item id="-248" type="2">
  <title>Wachstum einer Bakterienkolonnie</title>
  <isactive>true</isactive>
  <stem>
  ../activemath/PreguntaActiveMath?mbaseId=mbase://LeAM_calculus/
  exercises_rateofchange/open_bacteria
  </stem>
  <responseslayout>1</responseslayout>
  <isselfcorrected>false</isselfcorrected>
  <iccparameters difficulty="9" discrimination="0.75" guessing="0.25"
                topicid="-237"/>
  <templatetype>0</templatetype>

  <responses>

    <response id="-249">
      <text>#75#100#</text>
      <feedback/>
    </response>

    <response id="-250">
      <text>#50#74#</text>
      <feedback/>
    </response>

    <response id="-251">
      <text>#25#49#</text>
      <feedback/>
    </response>

    <response id="-252">
      <text>#0#24#</text>
      <feedback/>
    </response>
  </responses>
```

```
  <evaluation>

    <right>
      <responseid>-249</responseid>
    </right>

    <wrong>
      <responseid>-250</responseid>
    </wrong>

    <wrong>
      <responseid>-251</responseid>
    </wrong>

    <wrong>
      <responseid>-252</responseid>
    </wrong>
  </evaluation>
  <language>deutsch</language>
  <isexternal>true</isexternal>
  <patterntype>2</patterntype>
</item>
```

Test information This part contains information such as the topics involved, the item selection, assessment and finalization criteria must be known. An example of a Siette's test on the whole subject (over the root topic) is shown below:

```
<test id="-262">
  <name>Test on derivatives</name>
  <time/>
  <description>
    Test on derivatives
  </description>
  <isactive>true</isactive>
  <nummaxitems>10</nummaxitems>
  <numminitems>0</numminitems>
  <numknowlevels>12</numknowlevels>
  <respshuffle>true</respshuffle>
  <knowledgedistrib>1</knowledgedistrib>
  <selectioncrit>1</selectioncrit>
  <finalizationcrit>1</finalizationcrit>
  <correctioncrit>1</correctioncrit>
  <cadence>0</cadence>
  <rerunnable>false</rerunnable>
  <testlet>0</testlet>
  <topicid>-261</topicid>
  <creationdate>2001-06-21</creationdate>
  </test>
```

Finally, once this file has been created, it will be passed to an off-line utility, that would create a subject in the Siette's knowledge base. Note that all elements shown before use negative

identifiers. Negative identifiers are used to indicate to Siette than all contents are new. If identifiers are not negative, Siette will assume that they correspond to elements previously stored, but which need to be updated.

9.2 Automatic Generation of S-QTI files

As mentioned before, `SietteAsignaturaProducer` utility extracts on demand a course (or book) from *MBase*, and generates a S-QTI-based XML file.

Siette just stores references to exercises, represented by means of its external items. An example of how exercises are included in a S-QTI file is shown below:

```
<item id="-34" type="2">
  <title>Which polynomial describes a mountain top?</title>
  <isactive>true</isactive>
  <stem>
    ../activemath/PreguntaActiveMath?mbaseId=mbase://LeAM_calculus/
    deriv/fib_model2_diff_hiking
  </stem>
  <responseslayout>1</responseslayout>
  <isselfcorrected>false</isselfcorrected>
  <iccparameters difficulty="2" discrimination="0.75" guessing="0.25"
                 topicid="-33"/>
  <templatetype>0</templatetype>
  <responses>
    <response id="-35">
      <text>#75#100#</text>
      <feedback/>
    </response>

    <response id="-36">
      <text>#50#74#</text>
      <feedback/>
    </response>

    <response id="-37">
      <text>#25#49#</text>
      <feedback/>
    </response>

    <response id="-38">
      <text>#0#24#</text>
      <feedback/>
    </response>
  </responses>

  <evaluation>

    <right>
      <responseid>-35</responseid>
    </right>

    <wrong>
```

```
      <responseid>-36</responseid>
    </wrong>

    <wrong>
      <responseid>-37</responseid>
    </wrong>

    <wrong>
      <responseid>-38</responseid>
    </wrong>
  </evaluation>

  <language>english</language>
  <isexternal>true</isexternal>
  <patterntype>2</patterntype>
</item>
```

An item only stores information about the *MBase* identifier of the exercise, in order to be able to call it externally, by means of the delegation procedure. Besides, the responses, only contain information about how to process the response given by LeActiveMath.

Once the content is exported to Siette, assessment sessions can be requested on any of the topics, even though no tests have been defined for them. This can be done by means of the *parameterized tests* offered by Siette. These tests include the same information as the other types of tests, but they do not include the topics on which assessment takes place. This information is passed on as a parameter when one of these tests is invoked. Thus, these versatile tests are configured dynamically.

To have `SietteAsignaturaProducer` available, it has been encapsulated into a Web service. Consequently, content extraction is carried out on demand.

10 Loose Coupling

10.1 Loose Coupling Requirements

The web applications that take part in the process of delegating a user and his/her browser from one server to the other can play the following roles:

- *Guide server*: this is the server where the student browser is before this scenario comes into play. This guide server has some reasons to wish to send the user's browser to the following server. These reasons are probably the result of some previous exchange of knowledge.
- *Activity server*: is the one that will serve the activity to the browser that has been delegated by the guide server. The term activity is used quite loosely to describe any sequence of web interactions.

In interactions between LeActiveMath and Siette, the first system will play mainly the role of guide server and Siette the role of activity server, when LeActiveMath requests an assessment session, asks for results, etc. Roles are changed only when Siette requests exercise presentation. In this case, Siette assumes the role of guide server and LeActiveMath the activity server role.

From the end-user (student) perspective, he/she should perceive that both servers are more or less the same. To this end, the following requirements must be satisfied when loosely coupled web-applications are integrated:

- *single sign-on*: the user with his/her browser should not need to re-authenticate when passing from one server to the other. Based on the trust between the servers and on authentication at the guide server, authentication should be deduced on the activity server.
  Note that it is unreasonable to expect that the first contact of a user with the web-application on the activity server will always directly start the activity. For example, several web-applications will ask registration questions in order to seed their user-models. In the case of the loose-coupling of LeActiveMath and Siette, for example, both systems need to ask registration questions in order to seed the respective student-models.

- *interface homogeneity*: the users would be cognitively loaded by the need for a translation between different terms in the two servers to denote the same concepts. The same applies to the graphical vocabulary of, for example, icons. It appears thus, reasonable, to expect an amount of similarity which is estimated not to be too heavy a load, for example, by the teachers of the servers.

- *keep-going*: a dead-end in the navigation course will take the user away from his current goal. Therefore the guide server should continue guiding after an activity is finished or indicate the achievement of a milestone.

- *privacy*: since we are talking about the process of exchanging information related to the user experience, personal-history, and/or credentials to enter domains, the user should be sure that his data is kept safely. This safety is generally guaranteed at each servers site but needs to be considered in the exchange between the two servers. We expect encrypted and authenticated communication to be sufficient and manageable for this purpose.

10.2 The Browser Delegation Scenario

In this section, the steps of the delegation scenario as a sequence of remote procedure calls implemented as web services, are described in detail. This sequence is depicted in figure 10.

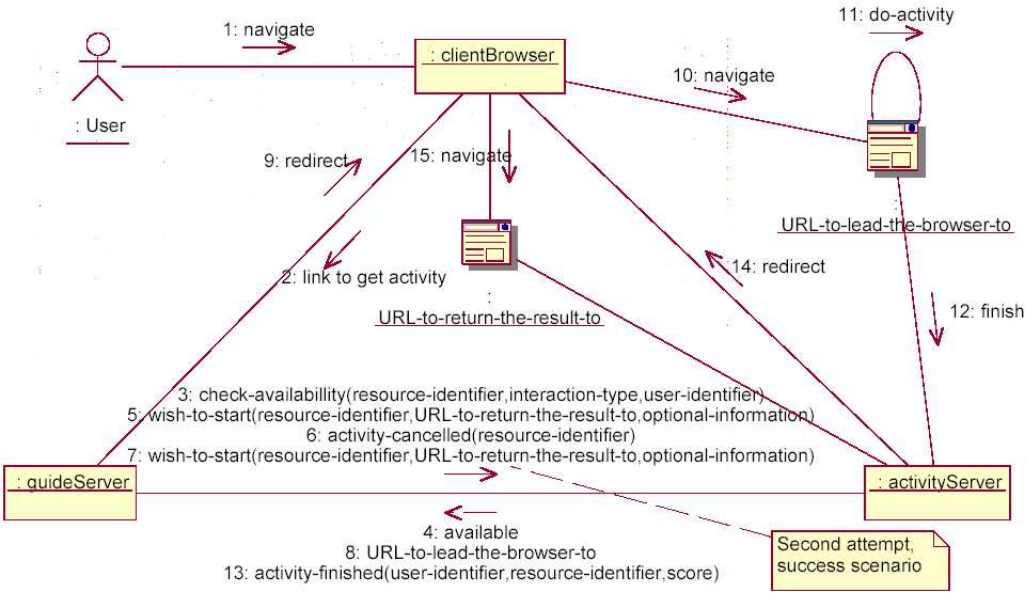

Figure 10: Loose-coupling steps.

The browser-delegation scenario starts when the student's browser is interacting with the *guide*

server. The latter, through authored content or discovery, offers a link that should lead the browser to the *activity server* for the time of the activity and *bring it back* when finished.

The scenario described here is a sequence of remote procedure calls to the guide or activity servers interleaved with browser actions. These calls are described below:

check-availability a call is made to check if delegation is possible. It includes:

- a *resource identifier*, i.e., a string that describes the activity univocally;
- an *interaction type* that represents an action describing what the interaction the browser is expected to have with the activity server. Examples include "start a test session", or "create a user". In the case of integration with Siette, these actions are implemented as Web services and will be approached in the following sections;
- a *user identifier*, i.e., a string to identify the user on both sides.

wish-to-start the guide server notifies the activity server, through a web service call, that it intends to send a browser to interact with a given resource. It provides at least the following parameters:

- the *resource identifier*, *interaction type*, and *user identifier* described before;
- other optional information to allow the interaction to be best suited to the user;
- a *URL-to-return-the-result-to* which is a resource locator to the web service to where it will invoke the *activity-finished* web service called later;
- optionally, a *URL-to-send-the-events-to* can be provided so that events can flow between the servers while the interaction happens.

In response to the notification received from the guide server, the activity server should provide a *URL-to-lead-the-browser-to*. The guide server now directs the browser to this URL. This URL should contain enough information so that the interaction can start right away.

activity-cancelled In some cases, the call to *wish-to-start* will not be followed by the actual activity. The guide server should then call to this method with the resource and user identifiers as parameters. The goal is to de-allocate any resource allocated for activities for this user and resource identifiers.

If the activity has not been cancelled, the student interacts with the resources on the activity server.

activity-finished When the interaction is finished, the activity server should direct the browser to future interactions that the guide server should indicate. The activity server contacts the *URL-to-return-the-result-to* and invokes an *activity-finished* method with parameters including the user and resource identifiers, as well as a numerical score, which should reflect the performance of the student during this interaction. This should be promptly answered by the guide server which will provide a *URL-to-come-back* which will be sent to the browser as the next place to go to.

11 Web Service-based Actions

Once the interaction scenarios have been described, this section approaches the actions offered by Siette when it plays the role of activity server. All these actions have been implemented using the popular technology of web services.

The previous experience with XML-RPC on LeActiveMath, where components are linked using this technology, and research on other technologies, resulted in the decision not to introduce a new way of communication in the LeActiveMath environment, but rather to use XML-RPC. The reasons are explained below:

- Initially, some new technologies, such as SOAP and WSDL, were considered. Although the current tendency is to use these protocols to implement web services, it would probably cause problems in the development of LeActiveMath. The introduction of an engine such as *Axis* [2] could have produced serious problems in debugging and development. Finally, the choice was to combine *REST* with XML-RPC. *REST* [8] is an architectural style, not a new web service implementation, but its use was thought to be both useful and advisable.
- Also, the easy integration of XML-RPC with Java code on both sides, is a point in favor of XML-RPC, and maybe against SOAP, which needs a little more effort as regards coding, using an external web service engine, etc.
- Simplicity and efficiency of XML-RPC is maybe the most important factor. The reduced bandwidth required for an XML-RPC call, makes it idoneous for a heterogeneous and distributed environment, such as LeActiveMath where components such as Siette might be deployed on different locations.

11.1 Services offered by Siette

The implementation of these services, described earlier in deliverable [18], entails the inclusion of a new layer, whose elements available the functionalities supplied by SIETTE available to LeActiveMath. These services are described below:

- `listOfTests (topicId,username,language)` → `list`

  Returns a list of available tests on a certain topic.

  Parameters: `topicId` is a topic identifier in Siette.
  A `username` to identify the student in Siette's knowledge base. It is useful to collect all tests to which the student is allowed to access.
  The `language` of the user, to locate the appropriate version of tests.

  Returns: A list of test identifiers.

- `startTestSession(testId,username,callback-URL,referenceId,student-model-initial-values)` → `URL-to-lead-the-browser-to`

  Start of the delegation scenario, where LeActiveMath passes the control to Siette for an assessment session.

  Parameters: A `testId` (obtained by a previous call to the service export-test, or maybe already stored).
  A `username` to identify the student in Siette's knowledge base.
  A `callback-URL` (XML-RPC endpoint) where Siette can send the result of the test.
  A `referenceId` to be passed to the callback, in order to identify in LeActiveMath the proper user/session with which the call is associated to a vector containing some optional

parameters to configure the test, according to the knowledge level of the student.
Example: `(topicId,initLevel,language,...)`

Returns: A string containing the URL for LeActiveMath to lead the browser to, when passing control to Siette.

- `createUser(username, password)` → `username`

  It creates the student model in Siette.

  Parameters: A `username` to be identified in Siette.
  A `password` (is the same in LeActiveMath in order to facilitate the delegation process).

  Returns: The username with which the student has been registered. This value can differ from the value passed on the parameter, since it could already exist.

- `getResultsOfPreviousTestSession(topicId,username)` → `list-of-marks`

  It returns the result obtained by a student in a given test, by querying on the topics involved in the test. Although [1] does not make any reference to it, this information should be used to update the LeActiveMath's student model.

  Parameters: A `topicId`, from a test for which LeActiveMath needs to obtain the assessment result.
  The `username` of the student who took the test.

  Returns: A list of pairs `(topicId,mark)` for each of the topics and subtopics involved.

- `summarizeItemUsage(itemId)` → `item-usage-list`

  Optional "teacher-oriented" function. Returns the results obtained by all the students that had taken this item as a part of a test.

  Parameters: An identifier of the item which the teacher needs to evaluate, calibrate, observe, etc.

  Returns: The distribution in discrete values `(final-mark-obtained-by-student,frequency)`.

  Example: (0,0.12);(1,0.35);...(11,0.07) The marks correspond to the knowledge levels of the test.

- `exportTest(siette-domain-knowledge)` → `testId`

  When domain knowledge must be updated in Siette, `SietteAsignaturaProducer` tool is invoked. As a result, it generates an S-QTI file. This file is passed on as a parameter to this service in order to accomplish the content update in Siette. In addition, if necessary, `SietteAsignaturaProducer` exporter could be invoked inside this service, and create a parameterized test.

  Parameters: `siette-domain-knowledge` is the S-QTI file.

  Returns: a testId, that is created, if necessary, as a parameterized test.

- `deleteTest(testId)`

  Deletes or disables the test. If the test was used, it cannot be deleted, but disabled, in order to keep the student results (this affects tests, items, statistics, etc.).

  Parameters: The identifier of the test to be deleted.

11.2 Events Issued by Siette

In the framework of loose coupling defined on LeActiveMath, events usually flow in the opposite direction to services, i.e. from LeActiveMath to Siette. Through events components inform LeActiveMath about their status. In Siette even just one might be considered:

- `testFinished`: this event is used to inform LeActiveMath that an assessment session has been completed. The `referenceId` passed as an argument to `startTestSession` helps on invoking the callback-URL by this event, to identify the session that a user was doing. Then, `getResultOfPreviousTestSession` is invoked to obtain the results of the assessment.

  This event is invoked by an `activityFinished` procedure call.

11.3 A Generic Case of Use

To sum up, this section is devoted to a case model that describes all the interactions between LeActiveMath and Siette.

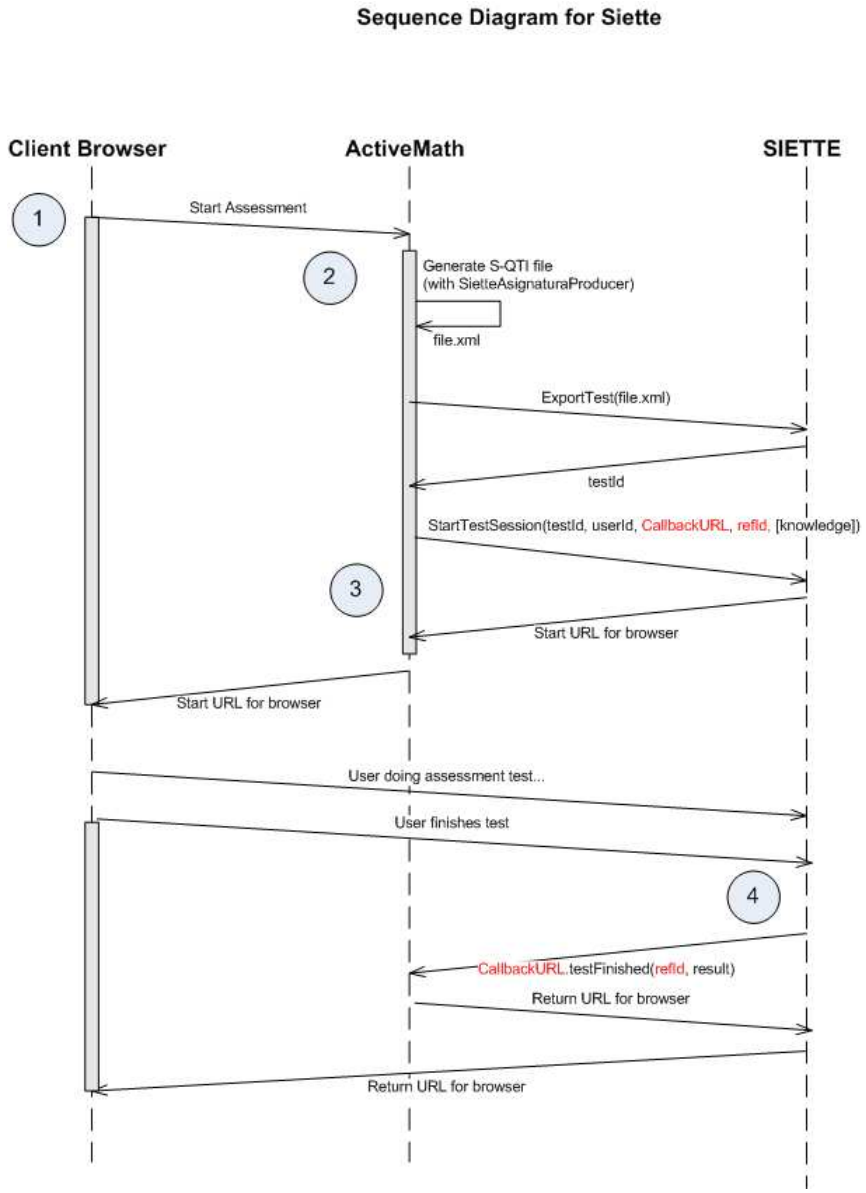

Figure 11: Sequence diagram of the interactions between Siette and LeActiveMath.

As mentioned before, the main integration scenario consists in requesting an assessment test

session from LeActiveMath to Siette. This will usually happen in the context of a book, when a student clicks on a link to start a test.

At this point, LeActiveMath starts the test session and gets the URL where the browser will be sent to. The student will be redirected to Siette and will click through the assessment.

When the assessment is finished, Siette will inform LeActiveMath by calling the callback. It will pass the reference id (to know to which user/session this call is associated to) and the test result. Also, the callback can contain a method to indicate failure/premature end of the test, etc. The callback returns a URL to Siette, which is where the user should be redirected to.

This scenario is described in figure 11, but there are two issues that have not been represented: Siette could make calls to `SietteAsignaturaProducer` as a service, for instance, to know if a test is already available, or to check the content stored against the content in LeActiveMath. Additionally, the call to `getResultsOfPreviousTestSession` done by LeActiveMath when the assessment is finished, gives not only the overall result on the test, but the results in each one of the topics involved in it.

12 Future Work

Siette offers other services that were not planned initially. The result analyzer can give information about sessions done by students on a subject (a book in LeActiveMath), sessions done by the same student, the difficulty, guessing and other parameters of items (exercises in LeActiveMath), statistics per item, groups of items, or a test, etc. These services could be offered by Siette and implemented as XML-RPC services, in the same manner as other simple activities planned in [18] have been.

Finally, the item calibration tool of Siette, although still under development, would be useful to calibrate the real item assessment psychometric features (e.g. difficulty), and other parameters of exercises of LeActiveMath. This could be an interesting future interaction scenario.

Part III

Description of the S-QTI Schema File

This appendix is a brief description of the XML schema[1] used to validate S-QTI files. Each document generated from this specification represents a whole knowledge base of a subject. The main element of this schema is the subject, whose type is the `type_subject` element.

13 `type_subject` type

Description: It is used to store all the subject properties, its domain model, its item pool and its test specifications.

```
<xs:complexType name="type_subject">
    <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="filespath" type="xs:string" minOccurs="0"/>
        <xs:element name="numknowlevels" type="xs:integer" minOccurs="0"/>
        <xs:element name="status" type="xs:integer" minOccurs="0"/>
        <xs:element name="isactive" type="xs:boolean" minOccurs="0"/>
        <xs:element name="topics" type="list_topics" minOccurs="0"/>
        <xs:element name="items" type="list_items" minOccurs="0"/>
        <xs:element name="tests" type="list_tests" minOccurs="0"/>
        <xs:element name="permissions" type="list_permissions"/>
        <xs:element name="topicid" type="xs:integer" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:integer" use="required"/>
</xs:complexType>
```

14 `list_topics` type

Description: It is composed of a list of the topics in the way they are ordered in the curriculum. The elements of this list are members of the `type_topic` element.

```
<xs:complexType name="list_topics">
    <xs:sequence>
        <xs:element name="topic" type="type_topic"
                    maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

15 `type_topic` type

Description: This type is used to store all the features of a topic and all its descendant subtopics. Figure 14 describes the elements that compose this type and figure 15 its attributes.

[1] This schema can be found at *http://www.lcc.uma.es/siette/docs*

Name	*Description*	*Use*
name	This is the name of the subject	mandatory
filespath	This is the relative path where the subject files will be stored.	optional
numknowlevels	Number of knowledge level in which students are going to be assessed. It takes the value 12 by default.	optional
status	It indicates the subject status with regard to the data stored in the database. By default, its value is 0. It means that the data have not been modified. Other possible values are: **1:** if data have been modified **2:** this subject (and all its contents) must be removed from the knowledge base. **3:** data are new.	optional
isactive	It is false if the subject is under construction (false). By default, it is true.	optional
topics	Set of topics of the subject	optional
items	List of items of the subject	optional
tests	Set of tests of the subject	optional
permissions	It contains the creator of the subject and a set of permissions given to other users	mandatory

Figure 12: Description of the elements of `type_subject`.

Name	*Description*	*Use*
id	Numerical identifier of the subject	mandatory

Figure 13: Description of the attributes of `type_subject`.

```
<xs:complexType name="type_topic">
    <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="status" type="xs:integer" minOccurs="0"/>
        <xs:element name="isactive" type="xs:boolean" minOccurs="0"/>
        <xs:element name="author" type="xs:string" minOccurs="0"/>
        <xs:element name="creationdate" type="xs:date" minOccurs="0"/>
        <xs:element name="modifiedby" type="xs:string" minOccurs="0"/>
        <xs:element name="modifieddate" type="xs:date" minOccurs="0"/>
        <xs:element name="translations" type="list_translations"
                    minOccurs="0"/>
        <xs:element name="subtopics" type="list_topics" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:integer" use="required"/>
</xs:complexType>
```

16 `list_translations` type

Description: It is composed of a list of topic name labels translated into different languages. The elements of this list are members of the `type_translation` type.

```
<xs:complexType name="list_translations">
    <xs:sequence>
        <xs:element name="translation" type="type_translation"
```

Name	Description	Use
name	This is the name of the topic	mandatory
status	It indicates the status of the topic with regard to the data stored in the knowledge base. By default, its value is 0. It means that the data have not been modified. Other possible values are: **1:** if data have been modified **2:** the topic (and all its subtopics and items) must be removed from the database. **3:** data are new	optional
isactive	It is false if the topic is already under construction (false). By default, it is true.	Optional
author	It is a valid username teacher in SIETTE. It corresponds to the creator of this topic.	optional
creationdate	This is the date of creation of the topic.	mandatory
modifiedby	This is the author of the last modification on the topic, if it has happened.	optional
modifieddate	This is the date of the last modification, if it has happened.	optional
translations	List of translations to other languages of the topic name.	optional
subtopics	List of descendants topics of this topic in the curriculum hierarchy	optional

Figure 14: Description of the elements of `type_topic`.

Name	Description	Use
Id	Numerical identifier of the topic	mandatory

Figure 15: Description of the attributes of `type_topic`.

```
                    maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

17 `type_translation` type

Description: This type stores a pair topic name/language. Through this type, internationalization of topics is expressed. Figure 16 describes the elements that compose this type.

```
<xs:complexType name="type_translation">
    <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="language" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
```

Name	Description	Use
name	This is the translation of the topic name into the corresponding language.	mandatory
language	This is a string with the language of this translation.	mandatory

Figure 16: Description of the elements of `type_translation`.

18 list_items type

Description: This type is used to collect the item pool of a topic. The elements of this list belong to the type_items type.

```
<xs:complexType name="list_items">
    <xs:sequence>
        <xs:element name="item" type="type_item" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

19 type_item type

Description: This type is used to store all the features and contents of an item. Figure 17 describes its elements and figure 18 its attributes.

```
<xs:complexType name="type_item">
    <xs:sequence>
        <xs:element name="title" type="xs:string"/>
        <xs:element name="isactive" type="xs:boolean" minOccurs="0"/>
        <xs:element name="status" type="xs:integer" minOccurs="0"/>
        <xs:element name="stem" type="xs:string"/>
        <xs:element name="responseslayout" type="xs:integer" minOccurs="0"/>
        <xs:element name="numrows" type="xs:integer" minOccurs="0"/>
        <xs:element name="numcolumns" type="xs:integer" minOccurs="0"/>
        <xs:element name="hint" type="xs:string" minOccurs="0"
                    maxOccurs="unbounded"/>
        <xs:element name="isselfcorrected" type="xs:boolean" minOccurs="0"/>
        <xs:element name="iccparameters" type="type_iccparameters"
                    maxOccurs="unbounded"/>
        <xs:element name="templatetype" type="xs:integer" minOccurs="0"/>
        <xs:element name="time" type="xs:string" minOccurs="0"/>
        <xs:element name="score" type="xs:float" minOccurs="0"/>
        <xs:element name="responses" type="list_responses"/>
        <xs:element name="evaluation" type="type_evaluation"/>
        <xs:element name="language" type="xs:string" minOccurs="0"/>
        <xs:element name="author" type="xs:string" minOccurs="0"/>
        <xs:element name="creationdate" type="xs:date" minOccurs="0"/>
        <xs:element name="modifiedby" type="xs:string" minOccurs="0"/>
        <xs:element name="modifieddate" type="xs:date" minOccurs="0"/>
        <xs:element name="note" type="xs:string" minOccurs="0"/>
        <xs:element name="keepresponseorder" type="xs:boolean"
                    minOccurs="0"/>
        <xs:element name="penalty" type="xs:float" minOccurs="0"/>
        <xs:element name="isexternal" type="xs:boolean" minOccurs="0"/>
        <xs:element name="patterntype" type="xs:integer" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:integer" use="required"/>
    <xs:attribute name="type" type="xs:integer" use="required"/>
</xs:complexType>
```

Name	*Description*	*Use*
title	This is the title which labels	mandatory
isactive	It indicates whether the item is ready to be included in tests	optional
status	It indicates the state of the item with respect to the data stored in the database. By default, its value is 0. It means that the data have not been modified. Other possible values are: 1: data have been modified 2: the item must be removed from the database.	optional
stem	This is the question of the item	mandatory
responseslayout	This is an integer variable that indicates the layout of the responses. There are three options: • **Sequential:** when responses are shown in a matrix of `(number_of_responses x 1)`. The variable takes the value 1. • **Lineal:** if responses are shown in a matrix of `(1 x number_of_ responses)`. The variable takes the value 2. • **Table:** if answers are shown in a matrix with the number of columns and rows equal to the values of the components *numcolums* and *numrows* respectively. The variable takes the value 3. By default it is sequential.	optional
numrows	Number of rows of the table layout of the responses. It only makes sense if the component *answerslayout* takes the value 3.	optional
numcolumns	Number of columns of the table layout of the answers. It only makes sense if the component *answerslayout* takes the value 3.	optional
hint	Optionally, users can provide the students with some hints to help them to solve the items.	optional
isselfcorrected	It indicates whether the item is a siettlet.	optional
iccparameters	It stores the psychometrical parameters of the item characteristic curve.	mandatory
templatetype	It indicates whether the item is a template for item generation. The value 0 (default option) indicates that this item is not a template. The remaining values indicate the language in which the template has been written. The values allowed are: 1. PHP 2. JSP 3. Perl	optional
time	Its format is *hh:mm:ss*. It is used to construct a timed item. That is, students will only have the time indicated available to answer.	optional
score	When tests' assessment criteria are "scored", this is the score if the student answers correctly	optional
responses	This is the set of possible responses to the item.	mandatory
Evaluation	This section indicates the way in which the item is assessed, and in addition, the type of item from the assessment point of view.	mandatory
language	This is the language in which the item is written. By default, it is assumed that it is written in Spanish.	optional
author	It is a teacher username valid in SIETTE. It is the item's owner.	optional
creationdate	This is the date of creation of the item	mandatory
modifiedby	This is the author of the last modification on the item, if modification took place.	optional
modifieddate	This is the date of the last modification, if modification took place.	optional
note	In this field, teachers can write some personal comment about the item. It is never shown to students.	optional
keepresponseorder	When tests are configured to show item answers in a disordered way, this value is used to force the item to keep the answer order.	optional
penalty	If a student fails to answer the item correctly, this value will be subtracted from his/her score.	optional
isexternal	It indicates whether it is an external item.	optional
patterntype	If it is an open-answer item, this value indicates the type of pattern	Optional

Figure 17: Description of the elements of `type_item`.

Name	Description	Use
id	Numerical identifier of the item.	mandatory
idcollection	Numerical identifier of the item collection, which are simply translations to different languages of the same item.	optional
type	It is an integer value that indicates the type of the item. Its possible values are: (1) *true/false*. (2) *multiple choice*. (3) *multiple response with independent answers*. (4) *multiple response with dependent answers*. (5) *open answer*.	mandatory

Figure 18: Description of the attributes of `type_item`.

20 `type_iccparameters` type

Description: This type includes the psychometric features of an item. These features, when they have been inferred after a calibration process, are essential in adaptive tests. Figure 19 describes the elements that compose this type.

```
<xs:complexType name="type_iccparameters">
    <xs:attribute name="difficulty" type="xs:integer" use="required"/>
    <xs:attribute name="discrimination" type="xs:float" use="optional"/>
    <xs:attribute name="guessing" type="xs:float" use="optional"/>
    <xs:attribute name="topicid" type="xs:integer" use="required"/>
</xs:complexType>
```

Name	Description	Use
difficulty	It is the difficulty of the item. It takes values from 0 to the number of knowledge level defined in the subject (*numknowlevels* element) minus one.	mandatory
discrimination	This value is the discrimination factor of the item.	optional
guessing	This probability suggests that a student without any knowledge will answer the item correctly and represents the case in which a student answers randomly.	optional
topicid	This value is a reference to a topic identifier. This means that this item can be used to assess the students' knowledge about this topic.	optional

Figure 19: Description of the attributes of `type_iccparameters`.

21 `list_responses` type

Description: This type is used to store the set of item response choices. Figure 20 describes the components of this type.

```
<xs:complexType name="list_responses">
    <xs:sequence>
        <xs:element name="response" type="type_response" minOccurs="2"
                    maxOccurs="unbounded"/>
        <xs:element name="blank" type="type_blank_response" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
```

Name	Description	Use
response	This is a possible answer to the item.	mandatory
blank	It represents the case in which a student does not select any answer.	optional

Figure 20: Description of the attributes of `list_responses`.

22 `type_response` type

Description: This type collects the information about an item response choice. Figure 21 describes the elements that compose this type, and figure 24 its attributes.

```
<xs:complexType name="type_response">
    <xs:sequence>
        <xs:element name="text" type="xs:string"/>
        <xs:element name="status" type="xs:integer" minOccurs="0"/>
        <xs:element name="feedback" type="xs:string" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:integer" use="required"/>
</xs:complexType>
```

Name	Description	Use
text	Text of the answer.	mandatory
status	It indicates the status of the answer regarding the data stored in the knowledge base. By default, its value is 0. It means that data have not been modified. Other possible values are: **1:** if data have been modified **2:** the answer must be removed from the database. **3:** data are new.	optional
feedback	It is a feedback that (in self-assessment tests and if student has selected this answer) is shown to the student within the item correction.	optional

Figure 21: Description of the elements of `type_response`.

Name	Description	Use
id	Numerical identifier of the answer	mandatory

Figure 22: Description of the attributes of `type_response`.

23 `type_blank_response` type

Description: This type is used to represent the case in which students do not select any item choice, i.e., leave the item blank. Figure 23 includes all its components and figure 24 its attributes.

```
<xs:complexType name="type_blank_response">
    <xs:sequence>
        <xs:element name="feedback" type="xs:string" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:integer" use="required"/>
</xs:complexType>
```

Name	Description	Use
feedback	It is a feedback that (in self-assessment tests and if student has selected this answer) is shown to the student within the item correction.	optional

Figure 23: Description of the elements of `type_blank_response`.

Name	Description	Use
id	Numerical identifier of the answer	mandatory

Figure 24: Description of the attributes of `type_blank_response`.

24 `type_evaluation` type

Description: This type is used to identify which item response choices are correct or incorrect. Figure 25 includes all its components.

```
<xs:complexType name="type_evaluation">
    <xs:sequence>
        <xs:element name="right" type="type_eval_response" maxOccurs="unbounded"/>
        <xs:element name="wrong" type="type_eval_response" minOccurs="0"
                    maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

Name	Description	Use
right	This indicates a correct answer.	mandatory
wrong	This element is not necessary, since by default, all remaining answers are considered incorrect.	optional

Figure 25: Description of the elements of `type_evaluation`.

25 `type_eval_response` type

Description: This type is used to represent the psychometric features of an item response. Figure 26 includes all the elements that compose this type.

```
<xs:complexType name="type_eval_response">
    <xs:sequence>
        <xs:element name="responseid" type="xs:integer"
                    maxOccurs="unbounded"/>
        <xs:element name="iccparameters" type="type_iccparameters"
                    minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="feedback" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
```

Name	*Description*	*Use*
responseid	This is the answer identifier. It corresponds to an `id` attribute of a `type_response` element.	mandatory
iccparameters	It stores the characteristic curve parameters corresponding to the response. This section can be omitted, if these parameters must be inferred from the item `iccparameters`.	optional
feedback	In addition to the feedback provided by each answer, a feedback for an answer combination can be given. It only makes sense in the case of *multiple response items with dependent choices*, since in this case, the correct response is a combination of choices.	optional

Figure 26: Description of the elements of `type_eval_response`.

26 `list_tests` type

Description: This type represents the list of tests of a subject. Their elements belong to `type_test` type.

```
<xs:complexType name="list_tests">
    <xs:sequence>
        <xs:element name="test" type="type_test" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

27 `type_test` type

Description: This type is used to store all the features of a test. Figure 27 includes all its components and figure 28 its attributes.

```
<xs:complexType name="type_test">
    <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="status" type="xs:integer"
                    minOccurs="0"/>
        <xs:element name="time" type="xs:string" minOccurs="0"/>
        <xs:element name="description" type="xs:string"
                    minOccurs="0"/>
        <xs:element name="isactive" type="xs:boolean"
                    minOccurs="0"/>
        <xs:element name="nummaxitems" type="xs:integer"/>
        <xs:element name="numminitems" type="xs:integer"/>
        <xs:element name="numknowlevels" type="xs:integer"/>
        <xs:element name="respshuffle" type="xs:boolean"
                    minOccurs="0"/>
        <xs:element name="knowledgedistrib" type="xs:integer"
                    minOccurs="0"/>
        <xs:element name="selectioncrit" type="xs:integer"/>
        <xs:element name="finalizationcrit" type="xs:integer"/>
        <xs:element name="correctioncrit" type="xs:integer"/>
        <xs:element name="assessmentcrit" type="xs:integer"/>
```

Name	*Description*	*Use*
name	This is the name of the tests	mandatory
status	It indicates the status of the test with respect to the data stored in the knowledge base. By default, its value is 0. It means that data have not been modified. Other possible values are: **1:** if data have been modified **2:** the test must be removed from the knowledge base. **3:** data are new.	optional
time	Its format is *hh:mm:ss*. It is used to make timed test. That is, students will only have available the time indicated available to finish the test.	optional
description	It is a brief description about the contents of the test.	optional
isactive	If it is false, it indicates that the test is already under construction. By default, it is true	optional
nummaxitems	Maximum number of items that can be posed to student in a test session.	mandatory
numminitems	Minimum number of items that can be posed to student in a test session before finishing it.	mandatory
numknowlevels	Number of knowledge level of the test. It must be a value ranging from 2 to the number of knowledge levels of the subject minus one.	mandatory
respshuffle	It indicates whether the responses of the items of the test must be shuffled each time they are posed to students. By default, it is true.	optional
knowledgedistrib	It gives information about the shape of the initial student knowledge probability distribution curve. Currently there are the following possible values: **1**: if all knowledge levels have the same probability. This is the default option. **2**: if it exists, the last curve stored in his/her student model.	optional
selectioncrit	It indicates the item selection criterion applied in the test. The following values are allowed: (1) Random. (2) Adaptive. (3) Bayesian. (4) Ordered. (5) Difficulty-ordered. (6) Entropy. (7) Maximum information function.	mandatory
finalizationcrit	It indicates the test finalization criterion. The following values are allowed: (0) Minimum probability. (1) Maximum reachable precision. (2) Maximum item number.	mandatory
correctioncrit	It indicates when the item corrections must be shown. There are the following possible values: (1) Immediately after posing the item. (2) At the end of the test. (3) Never.	mandatory
assessmentcrit	It indicates the assessment criterion applied in the test. There are the following possible values: (0) MAP. (1) EAP. (2) Percentage. (3) Score.	mandatory
cadence	This is the time in minutes that a student must await from the last test session, before being allowed to take it again. By default, the	optional

Figure 27: Description of the elements of `type_test`.

```
            <xs:element name="cadence" type="xs:integer"
                        minOccurs="0"/>
            <xs:element name="rerunnable" type="xs:boolean"
                        minOccurs="0"/>
            <xs:element name="testlet" type="xs:integer" minOccurs="0"/>
            <xs:element name="topicid" type="xs:integer"
                        maxOccurs="unbounded"/>
            <xs:element name="author" type="xs:string" minOccurs="0"/>
            <xs:element name="creationdate" type="xs:date"/>
            <xs:element name="modifiedby" type="xs:string" minOccurs="0"/>
            <xs:element name="modifieddate" type="xs:date" minOccurs="0"/>
            <xs:element name="showhints" type="xs:boolean" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:integer" use="required"/>
    </xs:complexType>
```

Name	*Description*	*Use*
id	Identifier of the test	mandatory

Figure 28: Description of the attributes of `type_test`.

28 `list_permissions` type

Description: This type is used to indicate who the author of the subject is, and to express a list of permissions on the subject granted to other users. Figure 29 describes all its components.

```
<xs:complexType name="list_permissions">
    <xs:sequence>
        <xs:element name="author" type="xs:string"/>
        <xs:element name="teacher" type="permissions" minOccurs="0"
                    maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

Name	*Description*	*Use*
author	This is the identifier of the user who has created the subject. He/she will have all the permission on creation, deletion and modification of the subject elements.	mandatory
teacher	This is a set of permissions that the author grants to other users. By default, only the author has all permissions granted to him/her.	optional

Figure 29: Description of the elements of `list_permissions`.

29 `permissions` type

Description: This type contains the permissions granted to a teacher on topics, items and tests of the subject. Figure 30 describes all the elements that compose this type.

```
<xs:complexType name="permissions">
    <xs:attribute name="teacher" type="xs:string" use="required"/>
    <xs:attribute name="topics" type="xs:integer" use="optional"/>
    <xs:attribute name="items" type="xs:integer" use="optional"/>
    <xs:attribute name="tests" type="xs:integer" use="optional"/>
</xs:complexType>
```

Name	*Description*	*Use*
teacherid	This is the identifier of the user who has been granted with the permissions.	mandatory
topicmodif	In this case, this user can modify the curriculum topics, but can neither create new topics nor delete them.	optional
topicdelet	In this case, this user can create, delete and modify the topics of the curriculum.	optional
itemmodif	In this case, this user can modify the curriculum items, but can neither create new items nor delete them.	optional
itemdelet	In this case, this user can create, delete and modify the curriculum items.	optional
testmodif	In this case, this user can modify the curriculum tests, but can neither create new ones nor delete them.	optional
testdelet	In this case, this user can create, delete and modify the curriculum items.	optional

Figure 30: Description of the elements of `permissions`.

References

[1] E. Andrès, P. Brna, N. Van Labeke, M. Mavrikis, R. Morales, H. Pain, and K. Porayska-Pomsta. Student model specification. deliverable d10. Technical report, LeActiveMath Consortium, December 2004.

[2] Apache Software Fundation. Axis. http://ws.apache.org/axis/.

[3] I. Arroyo, R. Conejo, E. Guzmán, and B. P. Woolf. An adaptive web-based component for cognitive ability estimation. In *Proceedings of the 9th World Conference of Artificial Intelligence and Education AIED'01*. Amsterdam: IOS Press, 2001.

[4] M. V. Belmonte, E. Guzmán, L. Mandow, E. Millán, and J. L. Pérez de la Cruz. Automatic generation of problems in web-based tutors. In L. C. Jain, R. J. Howlett, N. S. Ichalkaranje, and G. Tonfoni, editors, *Virtual Environments for Teaching and Learning*, Series on Innovative Intelligence, pages 237–281. London: World Scientific, 2002.

[5] S. Chua Abdullah. *Student Modelling by Adaptive Testing - A Knowledge-based Approach*. PhD thesis, University of Kent, Canterbury, June 2003.

[6] R. Conejo, E. Guzmán, E. Millán, M. Trella, J. L. Pérez de la Cruz, and A. Ríos. Siette: a web-based tool for adaptive testing. *Journal of Artificial Intelligence in Education*, 14:29–61, 2004.

[7] S. E. Embretson and S. P. Reise. *Item Response Theory for Psychologists*. Mahwah, NJ: Lawrence Erlbaum, 2000.

[8] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, http://www1.ics.uci.edu/ẽfielding/pubs/dissertation/rest_arch_style.htm, 2000.

[9] R. Flaugher. Item pools. In H. Wainer, editor, *Computerized Adaptive Testing: A Primer*, pages 51–65. Hillsdale, NJ: Lawrence Erlbaum Associates Publishers, 1990.

[10] E. Guzmán and R. Conejo. Simultaneous evaluation of multiple topics in siette. In S.A. Cerri, G. Gouardères, and F. Paraguacu, editors, *Proceedings of the 6th International Conference on Intelligent Tutoring Systems (ITS 2002). Lecture Notes in Computer Science*, number 2363, pages 739–748. New York: Springer Verlag, 2002.

[11] E. Guzmán and R. Conejo. A brief introduction to the new architecture of siette. In P. De Bra and W. Nejdl, editors, *Proceedings of the IIIth International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems(AH 2004). Lecture Notes in Computer Science*, number 3137, pages 405–408. New York: Springer Verlag, 2004.

[12] E. Guzmán and R. Conejo. A library of templates for exercise construction in an adaptive assessment system. *Technology, Instruction, Cognition and Learning (TICL)*, 2(1-2):21–43, 2004.

[13] E. Guzmán and R. Conejo. Self-assessment in a feasible adaptive web-based testing system. *IEEE Transactions on Education*, (in press), 2005.

[14] E. Guzmán and R. Conejo. Towards efficient item calibration in adaptive testing. In L. Ardissono, P. Brna, and A. Mitrovic, editors, *Proceedings of the 10th International Conference on User Modeling (UM 2005). Lecture Notes in Artificial Intelligence*, number 3538, pages 414–418. New York: Springer Verlag, 2005.

[15] E. Guzmán, R. Conejo, and E. García-Hervás. An authoring environment for adaptive testing. *Journal of Educational Technology & Society. Special Issue on Authoring*, 8(3), July 2005.

[16] R. K. Hambleton, J. Swaminathan, and H. J. Rogers. *Fundamentals of Item Response Theory.* Sage publications, 1991.

[17] LeActiveMath partners. Leactivemath: Annex 1: Description of work. Technical report, LeActiveMath Consortium, December 2003.

[18] P. Libbrecht, E. Andrès, O. Lemon, R. Morales, K. Porayska-Pomsta, and S. Winterstein. Open architecture. deliverable d8. Technical report, LeActiveMath Consortium, December 2004.

[19] C. G. Parshall, T. Davey, and P. J. Pashley. Innovate item types for computerized testing. In W. J. van der Linden and C. A. W. Glas, editors, *Computerized Adaptive Testing: Theory and Practice*, pages 129–148. Dordrecht (NL): Kluwer Academic Publishers, 2000.

[20] M. Scriven. The methodology of evaluation. In R. E. Stake, editor, *Perspectives of Curriculum Evaluation.* Chicago, IL: Rand McNally, 1967.

[21] H. Wainer. *Computerized Adaptive Testing: A Primer.* Lawrence Erlbaum, Hillsdale, NJ, 1990.