Implementation of an Assessment System Incorporating Web-based Parameterized Questions

doi:10.3991/ijet.v4i3.884

Vassilis Kapsalis
Technological Educational Institute (TEI) of Patras, Greece

***Abstract*—This paper presents a web-based assessment system, which supports parameterized questions, incorporating remote web services/applications, in order to use them during the evaluation process. The proposed system introduces a dynamic technique for creating a new type of parameterized web-based questions which mix static and dynamic content and thus, it operates as an integration framework, incorporating these remote web services/applications as part of the evaluation process. The technique is based on question templates, which contain the details of the remote services/applications that provide the correct answer in each question. The incorporation of remote web-based services into a parameterized assessment system, as part of the evaluation process, constitutes an advanced feature that provides new capabilities, such as flexibility and extensibility to any assessment system.**

***Index Terms*—Computer-assisted assessment, parameterized questions, web-based system, architectures for educational technology systems.**

I. INTRODUCTION

Web-based tests have become the primary tool for assessment and self-assessment of student knowledge in the context of web-based education and currently they constitute the most popular interactive component of all major web learning management systems (LMS) [1-3]. Also, there are several stand-alone web based assessment systems, such as Question Mark [4] and WebTest [5]. Their popularity is a result of the increasing class sizes [6-9] and the need for more efficient methods of assessing distant students [10-11]. These assessment tools support authoring and delivery of on-line tests, consisting of static type of questions, including a large variety of question types, such as multiple choice, multiple response, yes/no, fill-in-the-blank, essay, etc.

One of the main benefits from using online test systems is the capability for automated marking [12]. Although these systems work quite well in most of the cases, they feature certain restrictions related to the dynamic creation of questions. In order to address the problems of cheating and allow students the opportunity to redo tests, it is necessary to develop a large number of questions and apply a randomization of the questions of each test, in order to ensure that each test taken is sufficiently different. However, although this partially solves the problem of presenting different questions in each assessment session, it is rather a static solution, since all the available questions have to be preloaded in the database. Thus, if an author wishes to create new tests with similar questions, using simply different parameter values, he has to create these new questions or to properly modify existing ones by hand and subsequently to include these questions in the new tests. This is an important restriction regarding the flexibility of these systems. Furthermore, it requires a significant effort by the author in order to create and store similar questions at the database, because developing good quality questions takes considerable time [13]. Therefore, what is saved in marking is lost in development of a reasonably large number of quality questions [14].

Modern authoring tools enable the creation of web-based tests easier than it used to be. However, creating every question still involves a large amount of effort and time and subsequently the number of questions that can be created for a particular course is relatively small. In this context, it is quite natural, that in both assessment and self-assessment contexts all students in the class are using the same set of static questions.

Within an assessment, the lack of questions results in cycling through the same or nearly the same set of questions for each student in a class (and often for different classes and/or different semesters). This arrangement invites cheating, even in a controlled classroom context; and for take-home and web-based assessment, cheating becomes a major problem, often preventing teachers from relying on the results. In self-assessment where cheating is not a problem, the number of available questions is typically insufficient for students to assess the level of their knowledge or guide them in further study. To offset this problem, a method can be used whereby certain elements of the question are parameterized, in which randomly selected option(s) can be inserted to alter the details of the question, yet still testing the same content area [14-22]. Thus, creating parameterized questions can result in a large or infinite number of different test questions.

II. RELATED WORK

The main challenge regarding the use of parameterized questions and exercises as an educational technology is the evaluation of students' answers. A traditional static question can be evaluated by simply comparing the student's answer to a set of answers that have been provided by the author. However, a parameterized question requires a runtime application that can produce the correct answer for each instantiation of the question.

The use of parameterized questions into assessment systems constitutes an area of much research, and many prototype technologies have been developed [14], [15], [17], [22], [23]. The concept of parameterization lends

itself especially well to mathematical problems, where the answer is a number to be calculated. The idea is then to provide the student with a different set of (random) numbers each time the question is generated and to calculate the answer based on those numbers. While math is the most obvious field, Brusilovsky & Pathak [14] have also successfully developed a system suitable for use in teaching some topics in programming language courses.

The authoring of a parameterized quiz is not as homogenous as a typical static quiz and its implementation has considerable difficulty. Typically, it means that together with each question a simple program has to be provided that calculates the correct answer. This means that the author of a quiz must be familiar with a programming language. Another approach would be to make use of existing mathematics applications, like Mathematica or Matlab and invisibly use their computation capabilities by a server-side application in order to check the answers. An interesting concept is used by Maple T.A., in which, reusable assessment objects provide the capability for the randomization of everything from numbers, expressions and equations to graphs and diagrams, for creating tests and assignments and automatically assessing student performance [15]. In a similar way, in the case of programming languages it is possible to use an interpreter or compiler of the programming language in question to run the program provided by the student and compare the result with a program known to be correct [14].

Alternatively, the system could contain some kind of a general-purpose solver (often called a "domain expert"), that would obviate the need for a specific program to deal with each question. However, the difficulty of developing such a general-purpose solver is analogous to the degree of the generality that this solver has to provide. This is the main reason for which such systems are not commonly used.

In general, the existing parameterized systems are related on server-based application(s), each one being capable to evaluate parameterized questions of one specific type. Each question is being submitted to a web server, which, in turn, calls the corresponding application in order to produce the correct answer based on the question's parameter values and, subsequently, returns the result to the student. Thus, for each question, a simple connection gets established between a client application and the web server. A typical example implementing the above functionality, for assessing programming knowledge, is described in [14] and [24].

These parameterized assessment systems, although feature several dynamic characteristics, they, nevertheless, lack certain prevailing features of the Internet, such as the capability of incorporating and using the functionality of web-based remote services and applications. This missing functionality constitutes a major restriction of these systems, since the handling of each question instance is based on one or more local applications, which, usually, reside on the same server. However, the Internet includes a wealth of web-based tools, accessible through web pages and/or web services which may solve disparate problems, ranging from simple mathematical calculations to complex data processing. Thus, an assessment system, which could exploit distributed web services and applications through the Internet, would operate as an integration framework, incorporating these remote web services/applications as part of the evaluation process for parameterized questions. Thus, instead of using a local application for the evaluation of each question type, the system would exploit a number of distributed web-based services in order to achieve the evaluation task.

This new functionality constitutes the subject of this paper. Specifically, the paper presents an assessment system which supports parameterized questions, using remote web-based services during the evaluation process. The proposed system introduces a dynamic technique for creating a new type of parameterized questions which are based on web services/applications, mixing static and dynamic content.

This paper is organized as follows: In the next section, a description of the system functionality is presented, where the procedure for the manipulation of the main system items, such as the question templates, parameterized questions and tests, is described. Then, a complete scenario describing the run-time operation of the system is presented. Finally, the paper presents the conclusions and future work.

III. SYSTEM DESCRIPTION

Nowadays, a number of web pages (and web services) exist in the Internet, featuring either simple mathematical calculations or complex data processing performed by specialized application algorithms. In general, these web pages accept one or more input values and return a calculated result based on these inputs. Usually, their functionality is accessible through common web browsers. The innovation introduced by this paper is the incorporation of these remote web pages into the evaluation task of our assessment system. In order to achieve this, the concept of question templates is introduced. A question template is a high-level wrapper to a remote web page (or web service) that hides its presentation details and operates as an interface between the assessment system and the remote web page. Actually, it constitutes a means by which a communication path between the assessment system and the web page can be achieved, both for sending parameter values and for receiving the correct answer to each question.

The question templates are associated with parameterized questions. Each parameterized question includes the dynamic part (one or more placeholders), which, at presentation time, is instantiated with randomly generated parameters and the static part, which is the question's text. Actually, a parameterized question may be instantiated for as many times as has been configured in any test, in order to create the dynamic part of the test questions.

Each question template may be associated with one or more parameterized questions. During the run-time operation of each parameterized question, the system replaces the placeholders with their corresponding values and forwards the request at the proper service/application (URL endpoint). The response is (optionally) filtered by a filter which has been selected at the question template and is compared with the answer provided by the user. The comparison result is used for the user's assessment.

The support of parameterized questions based on remote web services/applications has been added to the functionality of a web-based assessment system, called WebTest [5], which has been developed, using the Active Server Pages (ASP) technology. It runs on the IIS web server, using either the MS Access or the MS SQL Server RDBMS. WebTest supports four types of questions: fill-

in-the-blank, multiple choice, multiple response and essay. Both the questions and the answers may contain text, images, sound and video content, supporting the most common file formats (doc, pdf, xls, txt, jpg, gif, wav, mp3, avi, mpeg, etc.). Each question may be contained in one or more tests and each test may be included in one or more lessons. Furthermore, a lesson may be contained in one or more courses. An author may create his own questions and, subsequently, may include them in his own test(s) or may select questions that have been already stored in the system by other authors. A student may assess his knowledge on the topics of any course by selecting any test of the specific course/lesson and answering to its questions. The system provides the capability for managing users, courses, lessons, tests, questions and their corresponding answers. Currently WebTest supports three courses at the Electrical Engineering Department of the Technological Education Institute (T.E.I.) of Patras: (a) computer programming, (b) principles of communications systems and (c) computer networks and integrated services. A UML use case diagram that illustrates the functionality provided by the WebTest system is shown in Fig. 1.

Figure 1. Use case diagram for WebTest

The new type of web-based parameterized questions that have been added to the system is the parameterized fill-in-the-blank (P-FIB) type. The creation of parameterized questions based on other question types is under development.

The functionality provided by the system, in order to manage question templates, filters and parameterized questions is illustrated in Fig. 2.

At the following sections, the procedure for the creation of a question template and its association with a number of parameterized questions, in order to create a test, is illustrated.

A. Creation of a question template

During the design phase the author creates a question template by filling the following fields at the corresponding web-based template:

- Name: The name of the template.
- Description: An optional description of the question template.
- URL address and the associated method (GET or POST): This field fills with the address of the web page, the HTTP method and a number of parameters that are used for transferring the values which are going to be used for the calculation of the correct answer. Each parameter is denoted as *@ParamValue[x]* and constitutes a placeholder that will be replaced by the actual value during the invocation of the question template by any associated question(s), which takes place during the run-time operation. Obviously, in order to fill this field, the tutor must know precisely the URL endpoint that will provide the correct answer.
- Extraction filter: A predefined function and the corresponding filter parameters for extracting the correct answer(s) from the returned html file. Actually, the function that implements the filtering action receives the response from the remote URL endpoint and (optionally) one or more filter parameters and produces (extracts) the output value(s). Again, in order to fill these fields, a precise knowledge of the returned html file is required.

As soon as the template has been created, any number of dynamic parameterized questions may be associated with it.

Figure 2. Use case diagram for the management of parameterized questions

B. Creation of a parameterized question

The procedure for the creation of a parameterized, fill-in-the-blank (P-FIB) question is as follows:

- Selection of a question template, which will be associated with this specific question.
- Filling both the static and the dynamic part of the question. Note that the dynamic part consists of one or more placeholders that correspond to the parameter(s) of the selected question template. During the run-time operation, these placeholders will be replaced by the proper values. This part of the question is denoted by the symbol(s): *@ParamValue[x]*, where x is an integer (x=1,2,…). The names and the number of these placeholders must match exactly to the names and the number of the parameters contained in the associated question template. Note that the label of the answer field constitutes part of the static part.

- Selection of a value range for each parameter value. Each parameter value may be a random number, may fall within certain limits, may be chosen from a predefined set of values or may be produced by a function. For example, a random integer value within the range [0,100] can be expressed as: Random [0,1,100].

Note that each parameterized question may be reused in one test any number of times, in contrast to a static question, which can be used only once per test.

C. *Creation of a test*

The tutor may write his own questions and, subsequently assign them to one or more tests or may create a test by selecting questions that have been previously stored in the database. Any parameterized question may be instantiated in a test any number of times. The number of instantiations of each parameterized question is configured during the test creation procedure. Each instantiation of a parameterized question creates a new set of parameter values, which, actually, may be considered as a new question.

D. *Run-time mode of operation*

The whole procedure that is followed from the moment that a user selects a question, until the evaluation of his answer takes place is the following: Whenever a dynamic parameterized question, which is based on a question template, executes, the system fills any question parameters in the corresponding placeholders (dynamic part of the question) with their corresponding values, which may be restricted within certain limits. Then, for each instance of a dynamic parameterized question, the system builds the proper request, as configured at the associated question template, and forwards it to the endpoint service/application that is responsible for the production of the correct answer. The answer provided by the remote endpoint is collected and parsed by the corresponding filter. In order to extract the useful information from the returned responses, a number of filtering functions have been implemented, that allow for the manipulation of HTML files. These functions provide the proper functionality for extracting values out of paragraphs, table fields, strings, etc. and, in general, manipulating most of the HTML tags. For XML content, these functions correspond to XPath expressions that can extract elements and/or attributes out of XML structures. Each of these extracted answers is finally compared with the corresponding student's answer. These comparisons produce the assessment result for each answer (correct or wrong), and finally, this result may be (optionally) presented to the user as feedback.

In order to present an insight of the assessment system, a detailed application scenario is described, in which a student is going to take a test related to IP subnetting. The test questions incorporate two web pages, which, are usually used as stand-alone network calculation tools. First of all, an analysis of the functionality provided by each of these web pages is necessary, in order to gather enough information for the creation of the corresponding templates and questions. The address of the first web page, illustrated in Fig. 3, is: "http://jodies.de/ipcalc".

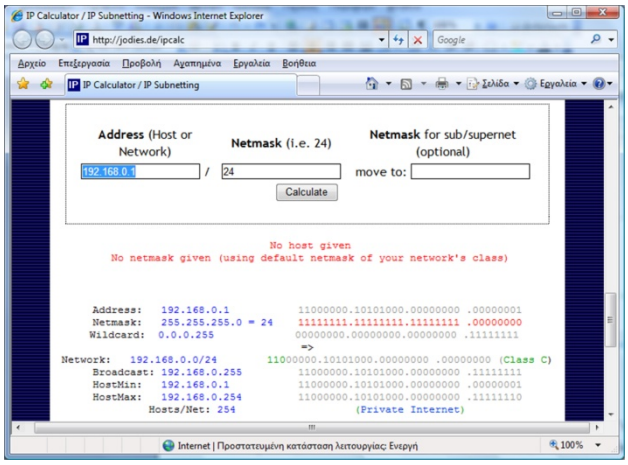

Figure 3. The web page "jodies.de/ipcalc"

Any user may type an IP address in the "*Address*" textbox (field: "*host*"), a netmask in the "*Netmask*" textbox (field: "*mask2*") and, optionally, the default mask for the class of the network, in the second "*Netmask*" textbox (field: "*mask1*") field. The page uses the HTTP GET method in order to read the values of these three fields and calculates for each subnetwork the following:

- network address
- first and last address
- broadcast address
- hosts/subnetwork

Also, the number of total subnetworks is calculated. Fig. 4 illustrates part of the page's html code, which includes the used method and the field names.

```
.
.
<form action="/ipcalc" method="get" name="form" id="form">
  <table border=0>
  <tr>
    <td><b>Address</b> (Host or Network)</td>
    <td><b>Netmask</b> (i.e. 24)</td>
    <td><b>Netmask</b> for sub/supernet (optional)</td>
  </tr>
  <tr>
  <td nowrap><input class=text name=host value="192.168.0.1"> / </td>
  <td nowrap><input class=text name=mask1 value="24"></td>
  <td nowrap> move to: <input class=text name=mask2 value=""></td>
  </tr>
  <tr>
    <td colspan=3>
    <input class=submit type="submit" value="Calculate">
    </td>
  </tr>
  </table>
  </form>
</div>
.
.
```

Figure 4. Part of the web page's html code

The web page, after receiving a request, executes the called operation and returns the result inside the following response (Fig. 5), which is actually an html file.

Observing the response html file, it is obvious that the number of calculated subnets is a value that resides between the strings: Subnets: <font color="#0000ff"> and <font color="#000000">
. The configuration for the question template (*Subnet_calculation*), illustrated in Fig. 6, is based on the information gathered from the first web page (jodies.de/ipcalc).

```
.
.
<font color="#000000">Network:  </font><font color="#0000ff">140.176.240.0/20    </font><font
color="#009900">10</font><font color="#909090">001100.10110000.</font><font color="#663366">1111
</font><font color="#909090">0000.00000000 (</font><font color="#009900">Class B<font
color="#000000">)</font><br><font color="#000000">Broadcast: </font><font
color="#0000ff">140.176.255.255     </font><font color="#909090">10001100.10110000.</font><font
color="#663366">1111 </font><font color="#909090">1111.11111111</font><br><font
color="#000000">HostMin:  </font><font color="#0000ff">140.176.240.1</font><font
color="#909090">10001100.10110000.</font><font color="#663366">1111 </font><font
color="#909090">0000.00000001</font><br><font color="#000000">HostMax:  </font><font
color="#0000ff">140.176.255.254</font><font color="#909090">10001100.10110000.</font><font
color="#663366">1111 </font><font color="#909090">1111.11111110</font><br><font
color="#000000">Hosts/Net: </font><font color="#0000ff">4094</font><br><br>
Subnets:  <font color="#0000ff">16 <font color="#000000"><br>Hosts:  <font
color="#0000ff">65504<font color="#000000"><br>
</pre>
<br>
.
.
```

Figure 5. Part of the response html code

The details of the question template are the following:

- Name: Subnet_calculation
- Description: An optional text that describes the functionality and the fields of the URL endpoint.
- URL: Since the actual URL endpoint which produces the correct answer is: *http://jodies.de/ipcalc?host=value1&mask1=value12 & mask2=value2*

  where *mask1* is optional, then the URL that must be entered in the corresponding field of the question template is:

  http://jodies.de/ipcalc?host=@ParamValue[1]&mask2=@ParamValue[2]

  where the *@ParamValue[x]* denote placeholders that will be replaced by their actual values during the invocation of the template by any associated question(s), which takes place during the run-time operation.
- Method: The HTTP method (GET) used by the web page.
- Filter: Since the result resides between two specific strings, the function that extracts the result is the "*GetTextBetweenText*".
- Parameters: The parameters used by the function "*GetTextBetweenText*" are: Subnets: <font color="#0000ff"> and <font color="#000000">
.

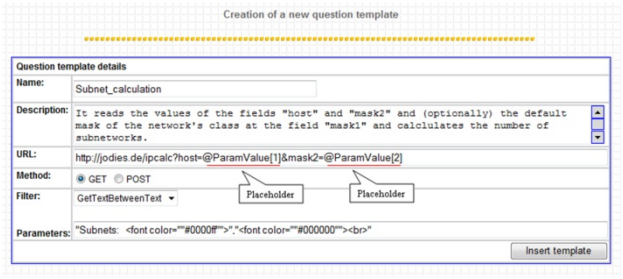

Figure 6. Creation of the question template: Subnet_calculation

The address of the second web page, illustrated in Fig. 7, is: "http://bighornent.com/cgi-bin/netmask-format.cgi". The functionality of this page is similar to that provided by the previous one. Specifically, it uses the HTTP GET method in order to read the address of a network (field: "*NETN*") and the applied mask (field: "*NETM*") and calculates the number of subnetworks and the hosts/subnetwork. Also, for a given number of subnetworks (field: "*NETS*"), it calculates the required netmask and the hosts/subnetwork.

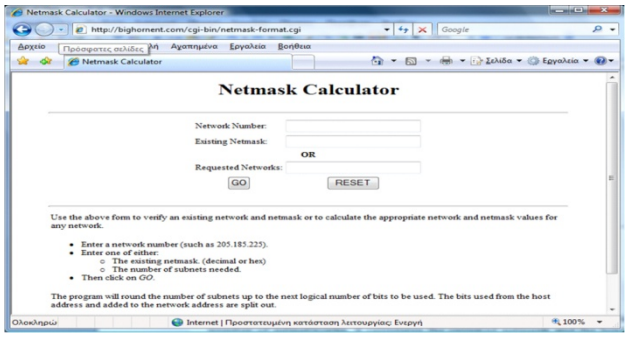

Figure 7. The web page "bighornent.com/cgi-bin/netmask-format.cgi"

Fig. 8 illustrates part of the page's html code, which includes the field names.

```
.
.
<FORM ACTION="netmask-format.cgi">
<TR><TD>Network Number:</TD><TD><INPUT NAME="NETN" SIZE=25
VALUE=""></TD>
<TR><TD>Existing Netmask:</TD><TD><INPUT NAME="NETM" SIZE=25
VALUE=""></TD>
<TR><TD COLSPAN=2 ALIGN=CENTER><B>OR</B></TD>
<TR><TD>Requested Networks:</TD><TD><INPUT NAME="NETS" SIZE=25
VALUE=""></TD>
<TR><TD ALIGN=CENTER><INPUT TYPE="SUBMIT" VALUE="GO"></TD>
</FORM>
<FORM ACTION="netmask-format.cgi">
<TD ALIGN=CENTER><INPUT TYPE="SUBMIT" VALUE="RESET"></TD>
</FORM>
.
.
```

Figure 8. Part of the web page's html code

The response is actually contained within a web page, which is illustrated in Fig. 9.

```
.
.
<TABLE BORDER=0 CELLPADDING=9>
<TR><TD><B>Net: 196.252.13.0</B><TD><B>Class: C</B><TD><B>Mask: 255.255.255.248</B></TR>
<TR>
<TR><TD><B>Bits Used: 5</B><TD><B>Subnets: 32</B><TD><B>Hosts/Net: 6</B></TR>
</TABLE><br>
.
.
```

Figure 9. Part of the response html code

Observing the response html file, it is obvious that the number of hosts/subnetwork resides between the strings: <B>Hosts/Net: and </B>. The configuration for the question template (*Hosts_per_subnet*), illustrated in Fig. 10, is based on the information gathered from the second web page (bighornent.com/cgi-bin/netmask-format.cgi).

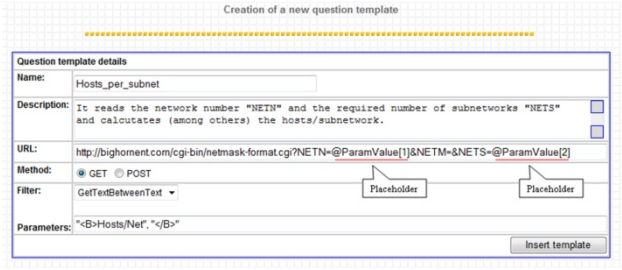

Figure 10. Creation of the question template: Hosts_per_subnet

The details of the question template are the following:

- Name: Hosts_per_subnet
- Description: An optional text that describes the functionality and the fields of the URL endpoint.
  - URL: *http://bighornent.com/cgi-bin/netmask-for-*

mat.cgi?NETN=@ParamValue[1]&NETM=&NETS=@ParamValue[2]

- Method: The HTTP method (GET) used by the web page.
- Filter: Since the result resides between two specific strings, the function that extracts the result is the "*GetTextBetweenText*".
- Parameters: The parameters used by the function "*GetTextBetweenText*" are the: <B>Hosts/Net and </B>.

The sample test contains four questions. The first two of them constitute instances of the same parameterized question, which uses the question template: *Subnet_calculation*. The third question, although it constitutes an instance of a different parameterized question, it, nevertheless, is associated with the question template that is used in the first two questions. The fourth question is based on a different parameterized question, which is associated with the *Hosts_per_subnet* question template. Fig. 11 illustrates the creation of a parameterized, fill-in-the-blank (P-FIB) question. Both the first and the second questions of the test constitute instances of this P-FIB question.

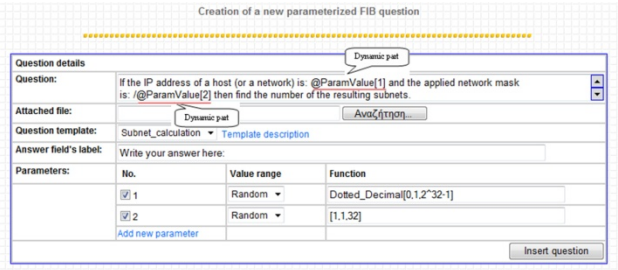

Figure 11. Creation of the first parameterized question

Analytically, the attributes of the first question, which constitute both the static and the dynamic part, are the following:

1st question

- Question: *If the IP address of a host (or a network) is: @ParamValue[1] and the applied network mask is: /@ParamValue[2] then find the number of the resulting subnets.*
- Answer's field label: *Write your answer here:*
- Static text: *"If the IP address of a host (or a network) is: ", " and the applied network mask is: /", " then find the number of the resulting subnets.", "Write your answer here:"*
- Parameters (dynamic part): *@ParamValue[1], @ParamValue[2]*
- Question template: *Subnet_calculation*
- URL endpoint: *http://jodies.de/ipcalc?host=@ParamValue[1]&mask2=@ParamValue[2]*

2nd question

It is assumed that the static part of the second question is the same with that of the first question, featuring identical attributes. Thus, both questions can be considered as instances of the same parameterized question which is illustrated in Fig. 11.

3rd question

The third question, although it constitutes an instance of a different parameterized question, it is associated with the same question template (*Subnet_calculation*) that is used in the first two questions.

- Question: *Calculate the number of subnets which are created, by applying the mask: /@ParamValue[2] at the IP address: @ParamValue[1].*
- Answer's field label: *Your answer:*
- Static text: *"Calculate the number of subnets which are created, by applying the mask: /", " at the IP address: ", ".", "Your answer:"*
- Parameters (dynamic part): *@ParamValue[1], @ParamValue[2]*
- Question template: *Subnet_calculation*
- URL endpoint: *http://jodies.de/ipcalc?host=@ParamValue[1]&mask2=@ParamValue[2]*

4th question

The fourth question is based on a different parameterized question, which is associated with the *Hosts_per_subnet* question template:

- Question: *Given the IP address @ParamValue[1] and the requirement to support @ParamValue[2] subnetworks, calculate the number of hosts/subnetwork alculate the number of subnets which are created, by applying the mask: /@ParamValue[2] at the IP address: @ParamValue[1].*
- Answer's field label: *Write your answer here:*
- Static text: *""Given the IP address ", " and the requirement to support ", " subnetworks, calculate the number of hosts/subnetwork.", "Write your answer here:""*
- Parameters (dynamic part): *@ParamValue[1], @ParamValue[2]*
- Question template: *Hosts_per_subnet*
- URL endpoint: *http://bighornent.com/cgi-bin/netmask-format.cgi?NETN=@ParamValue[1] &NETM=&NETS=@ParamValue[2]*

During the instantiation of the parameterized questions, a number of parameter values are produced, that fill their corresponding parameter placeholders, as illustrated in Table 1. Since the *@ParamValue[1]* corresponds to an IP address, the values that it may take are random numbers in the range [0, 2^{32}-1= 4.294.967.295], converted in dotted-decimal notation by the proper function: Dotted_Decimal[0, 1, 2^32-1] and the values for the *@ParamValue[2]* are random integer numbers in the range [1, 32].

TABLE I.
PARAMETER VALUES OF THE TEST QUESTIONS

Questions	@ParamValue[1]	@ParamValue[2]
1st Question	140.176.232.124	20
2nd Question	194.236.125.221	30
3rd Question	64.36.125.232	16
4th Question	196.252.13.224	32

The test which is presented at the student, after the instantiation of the parameterized questions is illustrated in Fig. 12. As soon as the student finishes typing his answers and clicks the "*Submit*" button, the system, for each question, constructs the corresponding request, as it has been configured at its associated template. For this specific example, the first three questions have been associated with the *Subnet_calculation* question template. This template uses the following URL endpoint: *http://jodies.de/ipcalc?host=@ParamValue[1]&mask2=@ParamValue[2]*. Thus, the system, after replacing the parameter placeholders at each one of the first three questions with their real values, as described earlier, creates three requests, which are transmitted to the same URL endpoint:

http://jodies.de/ipcalc?host=140.176.232.124&mask1=&mask2=20
http://jodies.de/ipcalc?host=194.236.125.221&mask1=&mask2=30
http://jodies.de/ipcalc?host=64.36.125.232&mask1=&mask2=16

The remote server, after receiving each request, executes the called operation and returns the result inside an html file. The filtering function extracts the result out of the html file. The final step is the comparison of the returned value with the student's answer, which will produce the assessment result (correct or wrong) that is used for the grading of the student. For the fourth question, the resulted request is the following: *http://bighornent.com/cgi-bin/netmask-format.cgi?NETN=196.252.13.224&NETM=&NETS=32*

The response from the remote server (bighornent.com) is filtered in a similar way by the corresponding function which has been configured at the associated question template, in order to produce the assessment result.

Fig. 13 illustrates the operations that take place during the execution of the above application scenario (the number inside the parenthesis indicates the time order of each operation). In general, the order of the received responses from the remote servers may be different at each test execution.

The UML sequence diagram of the above application scenario, which illustrates the transmitted messages between the client PC, the web server and the application servers, is illustrated in Fig. 14.

IV. CONCLUSION AND FUTURE WORK

The incorporation of remote URL endpoints into a parameterized assessment system, as part of the evaluation process, constitutes an advanced feature that adds new capabilities as well as flexibility and extensibility to any assessment system. The functionality of the parameterized questions, based on remote URL endpoints, has been incorporated in the new version of a web-based assessment tool, called WebTest. Currently, the system uses remote web pages that accept HTTP requests, carrying either GET or POST methods, with their corresponding parameters, which are incorporated into parameterized questions of fill-in-the-blank (FIB) type. The development of other type of parameterized questions (multiple choice, multiple response, etc.) is under investigation.

Figure 12. A sample test containing four parameterized questions

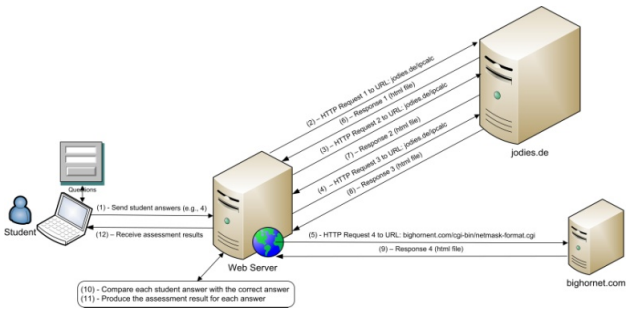

Figure 13. Time sequence of the run-time operations

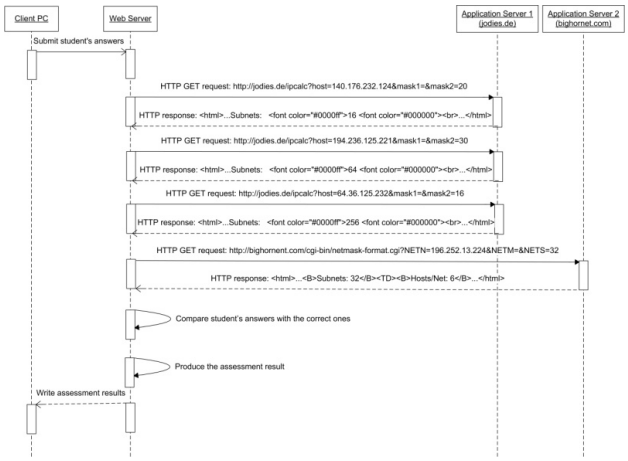

Figure 14. UML sequence diagram of the run-time operations

WebTest has been tested for the last year at three courses at the Electrical Engineering Department of the Technological Education Institute (T.E.I.) of Patras: (a) computer programming, (b) principles of communications systems and (c) computer networks and integrated services that have enrolments of over 400 students per semester. A number of web pages were created and incorporated into question templates, covering several topics of the above courses. The incorporation of remote web services/applications into each question has dramatically reduced the time needed for creating and updating test questions. For each topic, a small number of question templates produce a practically unlimited number of parameterized questions, ensuring that each test taken is sufficiently different. As a result, the students assess their knowledge in any topic by answering to a sufficient number of questions with different parameter values.

Due to the dependence of the assessment task on the responses of the remote web applications, the whole procedure takes more time to complete compared to static preloaded answers. However, this time, which is usually under one second, is hardly noticeable by the students and doesn't cause any practical problem to the assessment task. One case that needs to be handled is the lack of response by one or more web applications. A reasonable handling of this situation is the exclusion of any unanswered question from the assessment task and, at the same time, informing the student about the problem.

One of the features of the system is that, in order to incorporate remote web pages into the assessment system, an exact knowledge of the html code behind each page is required. This requirement poses a type of restriction, regarding the wide use of the system by tutors that have not expertise relevant to html and HTTP. Furthermore, since the extraction of the result is based on the web page's html code, it is prone to errors, since a modification of a web page's format may require a number of modifications of the filter's configuration. In order to simplify the whole process for non-technical tutors, a help file has been created for each incorporated question template, called template description (Fig. 11), which explains briefly its functionality (HTTP method, parameter names, etc.). In the next version of WebTest, a more advanced procedure is designed, in which, the system, during the creation of a question template, will parse each remote web page, in order to extract its input fields and, subsequently present them in a user-friendly way, ready for use by non HTML experts.

The above-mentioned disadvantages can partially be avoided, if the system provides the capability for the incorporation of web services, which are described by WSDL documents, instead of web pages. By using web services, the content is completely separated from the presentation. In this case, the communication is based on XML encoded SOAP messages, carrying web services operation calls and their corresponding responses. The extraction of the answer values out of the response messages may be achieved by using proper XPath expressions. Currently, a version that can support both web pages and web services is being designed.

Usually, quiz questions created in one quiz system may not necessary be in a compatible format to be re-used in another quiz system. To overcome this issue, the IMS Global Learning Consortium, that produces open specifications for interoperable learning technology has developed a specification for describing quiz questions. It has been designed such that all quiz system vendors who support the specification can import, export quiz question content, and allow sharing of quiz material. The IMS Question and Test Interoperability Specification (QTI) [25] is a complete and extensible specification, which supports many interoperable features as used by online quiz systems. However, features that are not included are: a) support for parameterized questions and b) incorporation of web services into the assessment system. A next step of our research is the study of issues related to the incorporation of parameterized questions with support of remote web services into a QTI-based assessment system.

REFERENCES

[1] Blackboard, "Blackboard learning system", Retrieved September 20, 2008 from http://www.blackboard.com/ClientCollateral/100506/Bb_Learning_System_%20Brochure.pdf.

[2] Claroline, "Claroline documentation", Retrieved September 15, 2008 from http://doc.claroline.net/en/index.php/Main_Page.

[3] Moodle, "Moodle course management system", Administrator documentation, Retrieved November 1, 2008 from http://docs.moodle.org/en/Administrator_documentation.

[4] QuestionMark, "QuestionMark perception authoring", Retrieved November 1, 2008 from http://www.questionmark.com/us/perception/FeatureComparisonAuthoring.pdf.

[5] Kapsalis, V., Simegiatou, V. & Fidas, C., "WebTest: An assessment system for education processes through the Web", in *3rd International Conference on Open and Distance Learning – ICOLD 2005* (pp. 618-628).

[6] Dalziel, J., "Integrating CAA with textbooks and question banks: Options for enhancing learning", in *Computer Aided Assessment* (CAA'2000).

[7] White, J., "Online testing: The dog sat on my keyboard", in *International Conference on Technology in Collegiate Mathematics*, 2000.

[8] Peat, M., Franklin, S. & Lewis A., "A review of the use of online self-assessment modules to enhance student learning outcomes: Are they worth the effort of production", in *ASCILITE'2001* (pp. 137-140).

[9] Pain, D. & Heron, J. L., "WebCT and online assessment: The best thing since SOAP?", *Journal of International Forum of Educational Technology & Society,* 6(2): 62-71, 2003.

[10] Sheard, J. & Carbone, A., "CADAL Quiz: Providing support for self-managed learning?", in *Proceedings of WebNet World Conference on the WWW and Internet 2000* (pp. 482-488).

[11] Hay, B. & Nance, K., "Using JavaScript for Client-side Online Testing", in *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications*, 2002, (pp. 1418-1419).

[12] Arnow, D. & Barshay, O., "On-line programming examinations using Web to teach", in *Proceedings of the 4th annual SIGCSE/SIGCUE conference on Innovation and technology in computer science education*, 1999, (pp. 21-24).

[13] Lister, R., "On blooming first year programming and its blooming assessment", in *Proceedings of the Australasian conference on Computing education*, 2000, (pp. 158-162).

[14] Brusilovsky, P. & Pathak, S., "Assessing Student Programming Knowledge with Web-based Dynamic Parameterized Quizzes", in *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications*, 2002, (pp. 1548-1553).

[15] Michael McCabe, "The search for extraterrestrial mathematics (Is there maths in the universe?)", in *CETL-MSOR Conference 2006*, pp. 113-119.

[16] Hubler, A. W. & Assad, A. M., "CyberProf: An intelligent human-computer interface for asynchronous wide area training and breakching", in *4th International World Wide Web Conference*, 1995.

[17] Kashy, E., Y. Thoennessen, Tsai, Y., Davis, N. E. & Wolfe, S. L., "Using networked tools to enhance student success rates in large

classes", in *Proceedings of the Frontiers in Education Conference*, 1997, (pp. 233-237).

[18] Merat, F. L. & Chung, D., "World wide web approach to teaching microprocessors", in *Frontiers in Education Conference*, 1997, (pp. 838-841).

[19] Woolf, B., Day, R., Botch, B., Vining, W. & Hart, D., "OWL: An Integrated Web-based Learning Environment", in *Proceedings of International Conference on Mathematics / Science Education and Technology*, 1999, (pp. 106-112).

[20] Lundquist, R., "Quiz collaboration -- cheating or a learning opportunity?", in *World Conference on Educational Multimedia, Hypermedia and Telecommunications*, 2001, (pp. 1177-1180).

[21] Ashton, H. S. & Beevers, C. E., "Extending flexibility in an existing online assessment system", in *Proc. 6th International CAA Conference (CAA'2002)*, 2002, (pp. 3-13).

[22] Clark D., "Enhancing the IMS Q&TI Specification by adding support for dynamically generated parameterised quizzes", Master's Thesis, University of Southern Queensland, 2004.

[23] Bridgeman, S., Goodrich, M. T., Kobourov, S. G. & Tamassia, R., "PILOT: An interactive tool for learning and grading", in *Proceedings of the 31st ACM SIGCSE Technical Symposium on Computer Science Education*, 2000, (pp. 139-143).

[24] Brusilovsky, P. & Sosnovsky, S., "Individualized exercises for self-assessment of programming knowledge: An evaluation of QUIZPACK", *ACM Journal of Educational Resources in Computing*, 2005, vol. 5, no 3, article 6, 1-22.

[25] IMS QTI, "IMS Question and Test Interoperability Overview", Version 2.1 Public Draft (revision 2) Specification, 8 June 2006, Retrieved 28 June 2009 from http://www.imsglobal.org/question/qtiv2p1pd2/imsqti_oviewv2p1pd2.html.

AUTHOR

Kapsalis is with the Electrical Engineering Department, Technological Educational Institute (TEI) of Patras, Greece (e-mail: kapsalis@ teipat.gr).

Submitted 25 March 2009. Published as resubmitted by the authors on 9 August 2009.