Developing A Public-domain Question Database for Physics Courses

Christopher D. Wentworth and Mark Plano Clark
Department of Physics
Doane College
Crete, Nebraska 68333

Evelyn T. Patterson and David E. Bell
Department of Physics
The United States Air Force Academy
USAF Academy, CO 80840-6254

Gregor M. Novak
Department of Physics
Indiana University-Purdue University Indianapolis
Indianapolis, Indiana, 46202

Abstract

College physics course environments, particularly at the introductory level, have become considerably more complex than a generation ago. The development of new information technologies and an expanding research base in physics education have produced dissatisfaction with a traditional teacher-centered, textbook driven curriculum. In particular, homework assignments need to reflect and enhance the total curriculum experience, not just what is in a textbook. Instructors need flexibility to try new kinds of questions quickly, get the benefit of rapid feedback from student performance, and share experiences with the wider physics education community.

These considerations have led us to develop a public domain question database with software which supports using the database in a web environment. Here we discuss issues in design philosophy, details of file specifications, an overview of the web delivery software, and some preliminary formative evaluation of using the database.

The design characteristics have been influenced by our desire to have a system that is easy to distribute to a variety of computer platforms and web-server environments, can be adapted easily as new question types are invented, and can make available an evolving collection of information about question use that can aid instructors in doing action research related to their use of homework assignments. These desired characteristics have led us to define the database using XML defined document types, with the ability to incorporate multimedia files and Java applets. Questions are created, edited, and assembled into web-based assignments using a software package written in Perl. Assignments can be classified into different categories: pre-class assignments, post-class homework, extra-credit, and quizzes. The database and delivery software will work on Unix, Windows NT, and Macintosh platforms.

Our preliminary data on using the database will help in developing an improved version. We will discuss the different kinds of web assignments which can be used and how they support the curriculum. Questions related to maintaining a distributed data base will be explored.

I. Introduction

We live in interesting times. Perhaps this is always true, and therefore, not a very profound statement, but for people involved in physics education two trends contribute to making these particularly interesting times. A large body of work in physics education is coming together with other work in the cognitive sciences to give us a valuable framework within which we can address questions of how best to teach and learn difficult physics concepts. At the same time, the development of web technology is giving us a greatly expanded ability to influence our students outside of the formal classroom. The convergence of these trends suggests an expanded set of opportunities for effective teaching. In this paper, we will be concerned with exploring the rationale and the practicality of some of the opportunities connected with web-based assignment systems.

At least since the 1980's, researchers in physics education have been concerned with what is really going on in people's minds when they are introduced to physics concepts. A reliable body of evidence on these thought processes has shown that there is divergence between what a teacher thinks is being communicated in an instructional context and the meaning that a student actually carries away from that experience.[1] Students begin a physics course with a model of the world already in mind, and instructional experience has too often left that model intact rather than moving the student forward toward a physicist's conception of things.[2]

Work in the cognitive sciences is beginning to give educators a theoretical framework for understanding some of the discoveries in physics education research and practical lessons for how to approach teaching physics.[3] Current cognitive research supports the idea that people do organize their experience into patterns or mental models: behaviorism is out. So a student is not a tabula rosa. An instructor should be aware of students' pre-conceptions and help students to become aware of them also. Another principle supported by cognitive studies is that our mental models control how we incorporate new information. As Edward Redish suggests[3]: we find that it is easy to learn something that matches or extends an existing model, or, to put it another way, "it is hard to learn something we do not almost already know".

One teaching strategy that results from these principles has been called the "given-new principle."[4] This strategy suggests that the new concept should be presented in a familiar context. We need to establish this context first and then introduce the new concept. Teachers are challenged to learn their students' mental models and establish a familiar context often with no real information other than instinct and perhaps years of experience in working with a particular clientele.

Another principle that comes from both physics education research and more general cognitive science research is that students can achieve the greatest development of their mental models of physics phenomena when instruction is carried out in a way that Richard Hake defines as "interactive engagement."[5] Such instructional methods are

> designed at least in part to promote conceptual understanding through interactive engagement of students in heads-on (always) and hands-on (usually) activities which yield immediate feedback through discussion with peers and/or instructors…[5]

The results of research in physics education and cognitive sciences have created dissatisfaction among many physics teachers. No longer are we satisfied with just explaining to our students the true nature of reality and assigning homework problems from one of a dozen mass produced textbooks that are difficult to distinguish from each other. We must construct learning environments where both teachers and students can gain understanding of the pre-existing mental models, where teachers can establish that familiar context in which learning best occurs, and where students are challenged to create new mental models by engaging in activities that promote individual reflection and peer/instructor dialogue.

Now it is time for web technology to step on stage. Whether it is coming from stage left, stage right, or directly from the audience, we do not know. But it is coming, and it offers an opportunity for establishing feedback loops between students and instructors that can help us understand our students' mental models before they get to class, build a context for learning, and provide feedback for applying new concepts in a quick and efficient manner. The technology also offers an opportunity to collect and share data on student thinking that can be the basis for practicing teachers to understand and modify their own teaching.

In the rest of this paper we will discuss one use of web technology that facilitates creating the learning ecology described above. The technology discussed here includes the creation, use, and sharing of a physics question database that allows web assignments which provide the needed feedback loops and support an interactive engagement environment. We will discuss different uses of web delivered questions that support the pedagogical framework mentioned above, design issues that determine the format of the database, and the supporting software which allows creating questions and assignments, delivering assignments to students, and sharing data between instructors.

II. Designing The Database

Our use of web-based assignments has very little to do with distance learning. We are using web-based assignments to build a more productive ecology for courses that meet in a formal classroom several times a week. Following the framework described above, instructors must try to understand pre-conceptions which students have before they arrive in class and then adjust the formal instructional time based on this data. Students also must be aware of their pre-conceptions to facilitate changing them. Instructors must try to establish a familiar context for students to share when they arrive at class. These pedagogical needs suggest that a constructive practice would be to have students do a pre-class assignment the results of which are available to the instructor before class so that they can be used for planning the class. Web technology allows this kind of assignment to be made and completed in the time constraints provided by the formal class. This kind of feedback loop between students and instructor has been called Just In Time Teaching.[6] The pre-class assignments are called "Warm-Ups" at Doane College and IUPUI and "Pre-flights" at the United States Air Force Academy.

The pedagogical framework described in the introduction also requires students to engage in "heads-on" activities that can lead to reflection and discussion. Traditional homework assignments often do not satisfy this criterion. Using computer-based multimedia capabilities such as digitized video or computer simulations allows building questions around a rich context that requires students to go far beyond looking up an equation in the textbook. Web technology allows us to deliver multimedia based questions in a platform independent fashion. This gives us an active learning alternative to textbook problems when we make our normal post-lesson homework assignments.

Using a web-based question database can also facilitate action research by instructors. If instructors could make an assignment and easily share the responses with other instructors then discussions might be initiated regarding what factors produced the best performance. A web-based question database offers the possibility of sharing performance results with people at many institutions. Institutions sharing the database could contribute new questions as well as performance data on existing questions. This offers a collaborative model for curriculum development.

Our collaboration for the design and production of the initial question database involved three rather different kinds of institutions: Doane College, a small liberal arts college, Indiana University-Purdue University – Indianapolis, an urban university, and the United States Air Force Academy. Not only are the institutions rather different educational environments, but the computing environments were also all different. Doane uses a Windows NT server, IUPUI uses a Macintosh server, and the USAFA uses a Unix web server. We needed a database and supporting software that could be used on all of these platforms with a variety of possible web server applications.

The need for serving this variety of environments pushed us to choose a flat (text) file database structure with supporting software written in Perl. This approach seemed to offer the most flexibility for constructing a system that could be used by everyone and that could be developed in a reasonable amount of time by physics professors who were not database experts. Flat files can be easily exchanged from one computing environment to another. Manipulating them can be achieved with relatively little cost in programming effort by using Perl, which is particularly well-suited as a programming language for doing character string manipulations.[7]

Another design characteristic of our database was that it had to be easy to add to the structure. In other words, we did not know with certainty all the kinds of information that should be included in our database. We did not even know all the kinds of questions that we wanted to use, because our expectation was that people would be creative and invent new question types. Clearly we needed to design the database so that it could evolve with a minimum of effort.

This design characteristic suggested that we use an XML[8] (extended mark-up language) document definition for our question database. XML is a simplified subset of SGML, and allows information providers to define their own tag and attributes for web documents, and to define an easily extendible document structure of arbitrary complexity. The language has been of particular interest to information providers who face the need to integrate different databases in a single application and to deliver information from databases over the web in an efficient manner. Documents written in XML are relatively easy for both humans and computers to read and interpret. We believed that defining an XML file structure for our question database would give us flexibility for adding to the structure in the future, and it would provide a common file format for merging questions from other databases which use proprietary file formats.

In addition to defining and creating a question database, we needed to provide a means of combining selected questions into assignments, delivering them to students, and storing the responses for later use. XML provided a means of defining the assignment database files and response database files.

III. Directory Structure and File Format

The question database itself is a flat file collection composed of one file for each question. These files are arranged in a directory structure that reflects a classification of physics subject matter known as the PIRA-X scheme. The PIRA-X scheme is an extended classification system devised at the United State Air Force Academy based on the PIRA scheme created by the American Association of Physics Teachers. Associated multimedia files are also classified by this scheme. So, the database itself contains the following sub-directories:

audios

classes

images

questions

videos

Each of these directories has nine sub-directories each corresponding to a major entry in the PIRA classification. The "classes" directory contains Java class files.

The filename for a question will be the PIRA-X code with an appended question number and a 'txt' extension. For example, a question on phonons could be contained in the file 4a60_001.txt. Institutions other than the USAFA will add an institutional code to the name. For example, at Doane College we are using the form

4a60_doane_001.txt

This should facilitate merging questions from different institutions.

The XML tags and attributes for the question files are defined in the Document Type Definition contained in Appendix 1. Here is an example of a question coded with the XML tags.

```
<question type="B" category="application" >
<topic code="1A40" name="Vectors"></topic>
<filename>
1A40_Doane_010.txt
</filename>
<creationDate>
19981016
</creationDate>
<version>
1
</version>
<text>
```

```
Velocity and acceleration are examples of a mathematical object called
a <_>.
</text>
<keyAnswer>vector</keyAnswer>
<instructorNote></instructorNote>
<author institution="Doane" name="Chris Wentworth"></author>
<keyWords>vectors</keyWords>
<history></history>
</question>
```

This question is a fill-in-the-blank type of question. The <text> tag contains the actual text of the question. This text could contain HTML code, if required. The “<_>” tag indicates where the blank will be located. The <keyAnswer> tag contains the required answer to be used by computer grading of the response. If there is a list of possible answers, each is separated by a comma. The <instructorNote> tag can contain notes about the question and its answer that could be seen by the instructor, but not by a student. The <keyAnswer> might be seen by the student, if the instructor chooses to use that kind of feedback. The <keyWords> tag can contain indexing information. The <history> tag is intended to contain results of using the question sent in by various institutions.

At this time, the system can accommodate essay, multiple choice, numerical, and fill-in-the-blank questions. The extendible nature of XML will allow us to add additional question types in the future.

We have a system for constructing, delivering, grading, and storing assignments that use the question database. We will describe it in the next section. This system requires that each course have an assignment file where questions associated with the assignment are specified. This file is also coded with an XML tag system. The document type definition is included in Appendix 2.

Here is an example of an assignment file.

```
<assignments>
<lesson number="1">
<name>
Warm Up #1
</name>
<instructions>
Please review chapter 3 in the textbook before trying this assignment.
</instructions>
<dueDate>
15:30-5-10-1998
</dueDate>
<keyAvailable value="never">
</keyAvailable>
<feedbackBefore type="name"></feedbackBefore>
<feedbackAfter type="summary"></feedbackAfter>
<randomMethod>
2
</randomMethod>
```

```
<questions>
<question points="5">
1A10_001.txt
</question>
<question points="5">1A10_002.txt
</question>
<question points="2">1A10_004.txt
</question>
</questions>
</lesson>
</assignments>
```

This particular class assignment file contains one web assignment in the <lesson> tag. Additional assignments could be included in the same file by using more <lesson>…</lesson> tags. Information that can be included within this tag includes the assignment title, special instructions to be printed at the beginning, the due date/time, whether the key can be made available to the student, what kind of feedback to give students after they submit their assignment, the way in which random numbers will be used, and the list of questions themselves. Notice that each question can have an instructor designated value, which is given in the "points" attribute of the question tag.

The assignment system described below stores student responses in a file that uses the tags defined in Appendix 3. Each course has a directory for its student response files. Each student has their own response file, in which all responses for assignments made in that course will be kept. Here is an example of a student response file.

```
<responses>
<lesson number="1">
<keyNew>
<qKey qName="Q_1" type="E" points="5">While the change in position and the change in time
between two events will generally depend on the reference frame in which the
position and time measurements are made, the spacetime interval does not
depend on the reference frame.  You always get the same number no matter
which (inertial) reference frame is used.  It is independent of the state
of motion of the observer.</qKey>
<qKey qName="Q_2" type="N" points="2">1.1,0.1</qKey>
<qKey qName="Q_3" type="MC" points="2"><mcAnswerKey
aType="D">meter</mcAnswerKey><mcAnswerKey
aType="D">second</mcAnswerKey><mcAnswerKey
aType="D">year</mcAnswerKey><mcAnswerKey aType="C">all of the
above</mcAnswerKey><mcAnswerKey aType="D">none of the above</mcAnswerKey></qKey>
seed=2
</keyNew>
<key>
<qKey qName="Q_1" type="E" points="5">While the change in position and the change in time
between
two events will generally depend on the reference frame in which the
```

```
position and time measurements are made, the spacetime interval does not
depend on the reference frame.  You always get the same number no matter
which (inertial) reference frame is used.  It is independent of the state
of motion of the observer.</qKey>
<qKey qName="Q_2" type="N" points="2">1.1,0.1</qKey>
<qKey qName="Q_3" type="MC" points="2">
<mcAnswerKey aType="D">meter</mcAnswerKey>
<mcAnswerKey aType="D">second</mcAnswerKey>
<mcAnswerKey aType="D">year</mcAnswerKey>
<mcAnswerKey aType="C">all of the above</mcAnswerKey>
<mcAnswerKey aType="D">none of the above</mcAnswerKey></qKey>
seed=2
</key>
<response qName="Q_1" score="5">Space & Time are really one thing
Motion is seperate from the observer
Do I know what this means? NO!</response>
<response qName="Q_2" score="2">1.15</response>
<response qName="Q_3" score="2">all of the above</response>
<startTime>Fri Jan 29 12:04:50 1999</startTime>
<endTime>Fri Jan 29 12:07:43 1999</endTime>
</lesson>
</responses>
```

This particular response file contains responses for one assignment. If the student had completed more assignments, the responses would be contained in the same file within another <lesson>…</lesson> tag.

IV. The Supporting Software

While developing the question database has been our primary interest, using the questions in real web assignments requires some kind of delivery and grading system that is compatible with the database. We have developed an assignment system that can be used with the question database to assemble web assignments from the database, create new questions, deliver assignments to students, grade some questions, and store all student responses for instructor viewing. The assignment system is called WebWorks and will be available at no cost to non-commercial users.

The system has been designed with the intention of providing multimedia-based questions (images, video clips, Java-based simulations) and mathematically oriented questions, in addition to more traditional multiple choice, fill-in-the-blank, and essay questions.

The code is written in Perl, with an expectation that it can be used on Unix, Macintosh, and Windows NT servers. The code draws on ideas developed by faculty at the USAFA, including David Bell, Dana Kopf, and Mike Hawks, in their pre-flight system, and by Larry Martin, author of WWWAssign.

The system consists of 6 Perl scripts, which handle creating, delivering, grading, and viewing assignments. Two modules contain common functions. There is a student web page, which serves as the student front-end to the system, and allows them to get a new assignment and view their submissions on previous assignments. There is an instructor web page, which serves as the instructor front-end to the system, that allows them to create assignments, view the student submissions, and to gade essay questions. The delivery, grading, and viewing scripts are named:

> webworkedit.pl—This script is called by the instructor page, and allows an assignment to be constructed from questions in the database or it allows the instructor to create new questions.
>
> webworkget.pl—This script is called by the student page and delivers an assignment to be submitted by the student.
>
> webworkgrade.pl—This script grades numerical, multiple choice, and fill-in-the-blank questions and stores the student responses for a submitted assignment.
>
> webworksview.pl—This script allows students to view previously submitted assignments.
>
> webworkiview.pl—This script allows instructors to view student responses. Instructors can choose to get a simple listing of all students in a class with their responses, or they can get a summary of student and class performance.
>
> webworkessay.pl—This script allows instructors to grade essay questions.

Students enter the system from the student web page. They select their class, section, assignment type and number, and enter their username and password. Submission of the form calls up the webworkget.pl script, which serves up the assignment. Submission of the assignment calls up the webworkgrade.pl script, which grades numerical, multiple choice, and fill-in-the-blank questions and stores all responses in the student’s response file.

The student can get several kinds of feedback, based on what the teacher selected when constructing the assignment. The system can send back a simple “Thank you” message, the total points earned for the assignment (not including essay questions), or a detailed listing of missed questions. The type of feedback will depend on the pedagogical intent of the teacher.

Instructors enter the system from the instructor web page. They select the class, section, assignment type and number, and enter their username and password. They can select to see a simple listing of all student responses from the class, see summary statistics for each student and the class on the particular assignment selected, grade essay questions, or create/edit an assignment.

Selecting existing questions or creating new questions are easy with the editing script. The teacher does not need to know HTML, although some knowledge of it can be helpful to add superscripts, subscripts, and other formatting details.

After accessing the editor a teacher must:

- pick a topic, organized by PIRA codes.

- look at existing questions in that topic; select an existing question for inclusion in an assignment, modify an existing question, or create a new question.

- when creating a new question, the author can provide an image (which is uploaded from the local machine and named with a name that reflects the question code assigned to this new question), question text, an answer, and other fields of interest such as comments to faculty who might use it. Each piece of information is provided via text areas and radio buttons in a web page viewed by the teacher.

- preview the question to see how it will look in a web-based assignment.

V. Formative Evaluation

We will look at a few observations on using the database.

The technical barriers of learning HTML, JavaScript, and server details can keep a faculty member from using web based questions. Being able to author questions and then letting the software store them in a database and deliver them to the students frees the faculty member to invest time in appropriately dealing with the student responses to those questions. The Just In Time Teaching philosophy requires that a teacher execute an extra step in preparing for class: review student responses to the warm-ups (or preflights). If the software is not close to being effortless to use then most faculty will not be able to fit the extra step into busy schedules.

Physics faculty at Doane College used the web delivery system without the editor for a semester. This experience showed that only a high level of commitment to the idea of web based assignments can overcome the difficulties of creating questions and assignments essentially from scratch. The editor is a crucial piece of the database system.

In the USAFA physics department, both of the introductory courses and a good number of the upper division courses now routinely use preflights, and this would not be the case were it not for the software that makes the implementation of preflights so easy. Other departments are taking the editor/delivery system and using it for their courses, too, so the impact of Just-in-Time Teaching is spreading beyond physics, largely because of the software that makes this implementation straight-forward.

Some of the student gains from using pre-class assignments (warm-ups or pre-flights) are described elsewhere.[6] Here we note some interesting student behaviors that may influence the success of using web based assignments.

At Doane we have used warm-ups in a pre-unit fashion, one warm-up before each unit (about once per week), and in a pre-class fashion, similar to the USAFA and IUPUI method. We believe the habit of mind created by doing the warm-up before every class is critical to seeing a major effect of this teaching strategy. When we did the warm-ups once per week, participation was never more than 80% of the class, and sometimes much less: mainly because students forgot about the assignment. The daily regimen helps create an expectation that students must do these assignments and facilitates them being prepared for class.

We have also given regular post-lesson homework assignments over the web. For these assignments, we have allowed students to get feedback on whether they answered non-essay questions correctly, and we have allowed re-submitted assignments. In this situation students will often make many submissions of the same question in an attempt to find the right answer. Based on the time stamps we can surmise that not a lot of thinking is going on between submissions. It is possible that giving this kind of feedback in the context of allowing re-submitted assignments is not very productive for students in a pedagogical sense.

VI. Conclusion

The question database and the assignment delivery system are evolving.. The next major step is too develop the best procedure for sharing new questions and performance data between institutions. We hope that this will produce collaborative curriculum development. Our experience shows that the assignment system itself is easy enough to use so that other academic departments have shown an interest in it. For example, the Geology Department at Doane College will be using the system during the fall 1999 semester, and the German Department has also expressed interest. The ability to use multimedia, such as audio files, in homework questions can be very powerful.

Further information on the question database and assignment delivery system can be obtained from the project web site.[10]

References

1. Arnold Arons, *A Guide To Introductory Physics Teaching*, John Wiley & Sons, Inc., New York (1990).

2. I.Halloun and D. Hestenes, “The initial knowledge state of college physics students,” Am. J. Phy. 53, 1043 (1985).

3. Edward F. Redish, “Implications of cognitive studies for teaching physics,” Am. J. Phys. 62, 796 (1994).

4. H. Clark and S. Haviland, “Comprehension and the given-new contract,” *in Discourse Production and Comprehension*, edited by R. Freedle, Lawrence Erlbaum Associates, Hillsdale, NJ (1975).

5. Richard R. Hake, “Interactive-engagement versus traditional methods: a six-thousand student survey of mechanics test data for introductory physics courses,” Am. J. Phys. 66, 64 (1998).

6. Gregor M. Novak, Evelyn T. Patterson, Andrew D. Gavrin, and Wolfgang Christian, “Just-in-Time Teaching: Blending Active Learning with Web Technology,” Prentice Hall, Upper Saddle River, NJ, (1999).

7. Larry Wall, Tom Christiansen, and Randal L. Schwartz, *Programming Perl*, O’Reilly & Associates, Inc., Sebastopol, CA (1996).

8. Norman Walsh, “A Guide to XML,” in *XML: Principles, Tools, and Techniques*, edited by Dan Connolly, Travelers’ Tales Inc. (1997).

9. Larry Martin, “WWWAssign,” http://www.northpark.edu/~martin/WWWAssign/ , North Park University, Chicago, IL. (1997).

10. “WebWorks Web Assignment Project,” http://www.doane.edu/crete/academic/science/phy/jitt/wwproject.htm , Doane College, Crete, NE.

Appendix 1. Document Type Definition For The Question Files

```
<!ELEMENT question (topic , filename , creationDate , version , image*, text ,
mcAnswer* , keyAnswer?, instructorNote ?, author?, keywords?, history? ) >
<!ELEMENT topic EMPTY>
<!ELEMENT filename (#PCDATA)>
<!ELEMENT creationDate (#PCDATA)>
<!ELEMENT image (#PCDATA)>
<!ELEMENT version (#PCDATA)>
<!ELEMENT text (#PCDATA)>
<!ELEMENT mcAnswer (#PCDATA)>
<!ELEMENT keyAnswer (#PCDATA)>
<!ELEMENT instructorNote (#PCDATA)>
<!ELEMENT author EMPTY>
<!ELEMENT keyWords (#PCDATA)>
<!ELEMENT history (#PCDATA)>
<!ATTLIST question
        type        (MC|E|B|N|S)            #required
        category    (application|puzzle|review|test|warmUp) ‘application’ >
<!ATTLIST mcAnswer
        correct     >
<!ATTLIST topic
        code        CDATA                   #required
        name        CDATA                   #required  >
<!ATTLIST  filename
        src         CDATA                   #required  >
<!ATTLIST image
        src         CDATA                   #required >
<!ATTLIST author
        institution CDATA ‘unknown’
        name        CDATA ‘unknown’>
```

Appendix 2. Document Type Definition For The Assignment File

```
<!ELEMENT assignments (lesson?)>
<!ELEMENT lesson (name, instructions, dueDate , keyAvailable, feedbackBefore,
feedbackAfter, randomMethod, questions) >
<!ELEMENT name (#PCDATA)>
<!ELEMENT instructions (#PCDATA)>
<!ELEMENT dueDate (#PCDATA)>
```

```
<!ELEMENT keyAvailable EMPTY>
<!ELEMENT feedbackBefore EMPTY>
<!ELEMENT feedbackAfter EMPTY>
<!ELEMENT randomMethod (#PCDATA)>
<!ELEMENT questions (question*)>
<!ELEMENT question (#PCDATA)>
<!ATTLIST lesson
        number    ID      #required >
<!ATTLIST keyAvailable
        value     (always|never|after) 'never' >
<!ATTLIST feedbackBefore
        type      (name | answers | summary | missed ) >
<!ATTLIST feedbackAfter
        type      (name | answers | summary | missed ) >
<!ATTLIST question
        points    CDATA '1'>
```

Appendix 3. Document Type Definition For The Student Response File

```
<!ELEMENT responses (lesson*) >
<!ELEMENT lesson (key, keyNew, response+) >
<!ELEMENT key (#PCDATA,qKey)>
<!ELEMENT qKey(#PCDATA)>
<!ELEMENT keyNew (#PCDATA,qKey)>
<!ELEMENT response (#PCDATA , comment* ) >
<!ELEMENT comment (#PCDATA) >
<!ATTLIST lesson
        number    ID      #required >
<!ATTLIST qKey
        qName     CDATA               #required
        type      (MC|E|B|N|S)        #required>
<!ATTLIST response
        qName     CDATA #required
        score     CDATA 'none' >
```