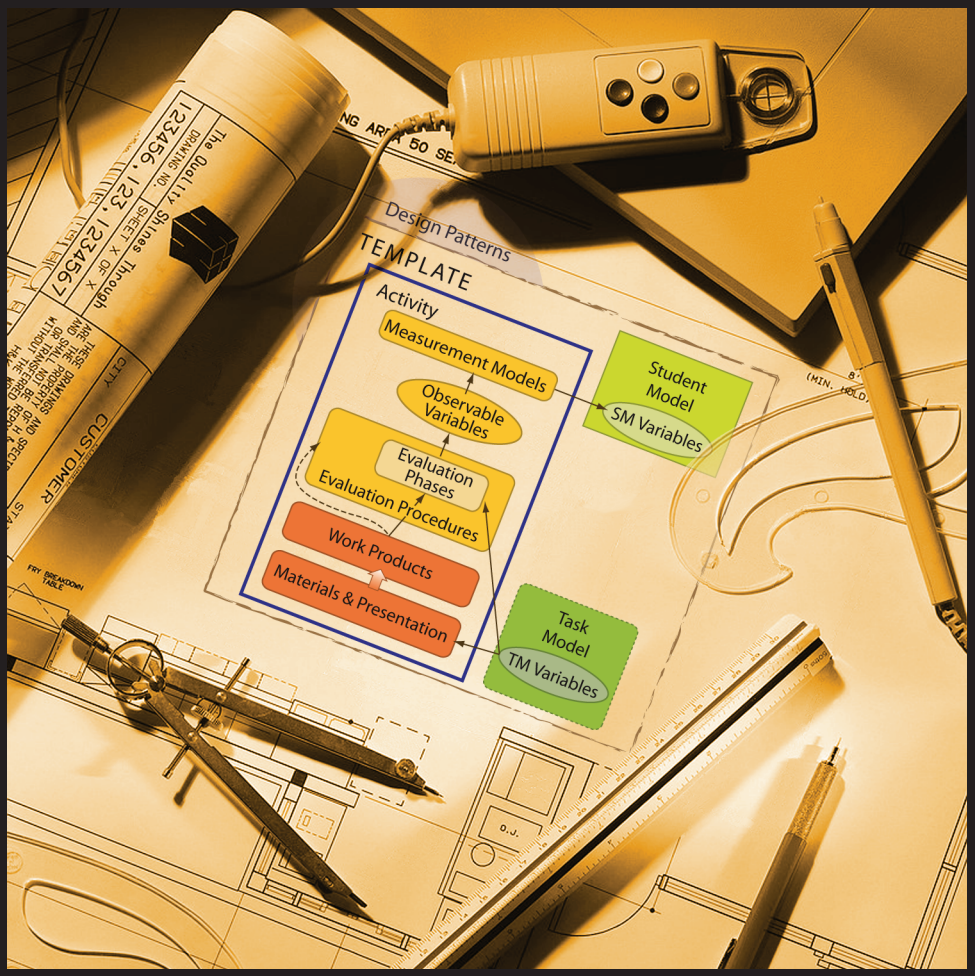


PADI Technical Report 12 | March 2006

A Guide to the PADI Gradebook

PADI | Principled Assessment Designs for Inquiry

Lawrence Hamel, CodeGuild, Inc.
Robert Mislevy, University of Maryland
Cathleen A. Kennedy, University of California at Berkeley

Report Series Published by SRI International

SRI International
Center for Technology in Learning
333 Ravenswood Avenue
Menlo Park, CA 94025-3493
650.859.2000
http://padi.sri.com

PADI Technical Report Series Editors
Alexis Mitman Colker, Ph.D., *Project Consultant*
Geneva D. Haertel, Ph.D., *Co-Principal Investigator*
Robert Mislevy, Ph.D., *Co-Principal Investigator*
Meredith Ittner and Klaus Krause, *Technical Writers/Editors*
Lynne Peck Theis, *Documentation Designer*

PRINCIPLED ASSESSMENT DESIGNS FOR INQUIRY
TECHNICAL REPORT 12

A Guide to the PADI Gradebook

Prepared by:
Lawrence Hamel, CodeGuild, Inc.
Robert Mislevy, University of Maryland
Cathleen A. Kennedy, University of California at Berkeley

Acknowledgments
This material is based on work supported by the National Science Foundation under grant REC-0129331 (PADI Implementation Grant).
Disclaimer
Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

FIGURES

ABSTRACT

The PADI Gradebook is an application created under the Principled Assessment Designs for Inquiry (PADI) project that combines assessment design information with student response data and employs a scoring engine to generate estimates of student proficiency. Gradebook supports viewing, editing, and importing of scores. It supports some simple, automatic evaluation actions on item scores that are part of a bundle of dependent items, if this bundling is specified in the assessment design. After using the scoring engine to calculate student proficiency, Gradebook displays a summary of student proficiencies in a graphical presentation.

1.0 Background

In the Principled Assessment Designs for Inquiry (PADI) project, the PADI design system provides a structure for developing assessment designs with clear rationales. This report assumes the reader is familiar with PADI concepts such as Student Model Variables, Observable Variables, Measurement Models, and the scoring engine, as they exist as part of the PADI project at the time this guide was composed (see Kennedy, 2005a, 2005b; Mislevy & Riconscente, 2005; Riconscente, Mislevy, Hamel, & PADI Research Group, 2005). Gradebook also draws on concepts from the four-process architecture[1] for assessment delivery (Almond, Steinberg, & Mislevy, 2002).

Robert Mislevy at the University of Maryland, one of the Principal Investigators of PADI, directed the creation of a prototype tool that would make use of assessment design information and the proficiency estimates that result from following such a design. Professor Mislevy teaches a course within the College of Education, Department of Measurement, Statistics & Evaluation (EDMS), and imagined a tool that would keep track of students, classes, and assessments. The resulting prototype is called the PADI Gradebook, referred to as simply "Gradebook" below. It is an example of a PADI application, a concrete deployment of some of the abstract concepts and data structures that are contained within the PADI design system.

The scoring engine used by Gradebook was created by a research group at the University of California, Berkeley, led by Mark Wilson. More information about this scoring engine (also referred to as the BEAR Scoring Engine) is available from the Web site of that research group (<http://bearcenter.berkeley.edu/>) and in PADI publications. This scoring engine supports multidimensional item response theory (IRT) scoring using models that can be expressed as instances of the multidimensional random coefficients multinomial logit model (MRCMLM; Adams, Wilson, & Wang, 1997), including the Rasch model for binary items, partial credit and rating scale models, and bundle models for conditionally dependent item responses. The BEAR Scoring Engine is a PADI application that implements the MRCMLM to produce estimates of student proficiencies. This model provides a generalized solution for a family of multidimensional, polytomous models to produce inferences about student knowledge or ability from the evidence represented by conditionally independent Observable Variables.

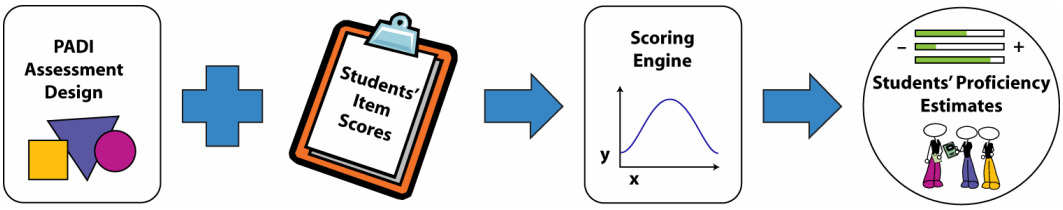
[1] The four-process architecture includes one process each for activity selection, presentation, evidence identification, and evidence accumulation. A central library serves to tie together information from each process. Examinees interact solely with the presentation process.

2.0 Purpose of Gradebook

Gradebook combines assessment design information with student scores and employs a scoring engine to generate estimates of student proficiency. Gradebook is not involved with the design, authoring, or delivery of assessments. It presupposes that a PADI assessment design has been created in some separate design-supplying system, and that the assessment design somehow has been translated into a structured collection of one or more actual assessment tasks that the instructor has delivered to a group of students. The assessment design consists of a collection of specifications of materials, procedures, and, of particular importance to Gradebook, parameters and supporting information for psychometric measurement models. The location of this collection of task-design models is specified in Gradebook with a Uniform Resource Locator (URL). Any system can supply the design information as long as it follows the prescribed PADI XML format, supplied via the Web's standard Hypertext Transfer Protocol (HTTP). At the time of writing, all experimentation has been done with a single design supplier, the PADI design system.

Thus, in the first step of the logical flow of information in the Gradebook process, the instructor specifies a URL for the assessment design models. Gradebook retrieves and stores the assessment design models from the design-supply system into the Gradebook database, as represented by the first box within Figure 1. This first box is labeled "PADI Assessment Design" and includes shapes to represent the assessment design models.

Figure 1. Purpose of Gradebook: To Combine Design Information with Item Scores to Produce Estimates



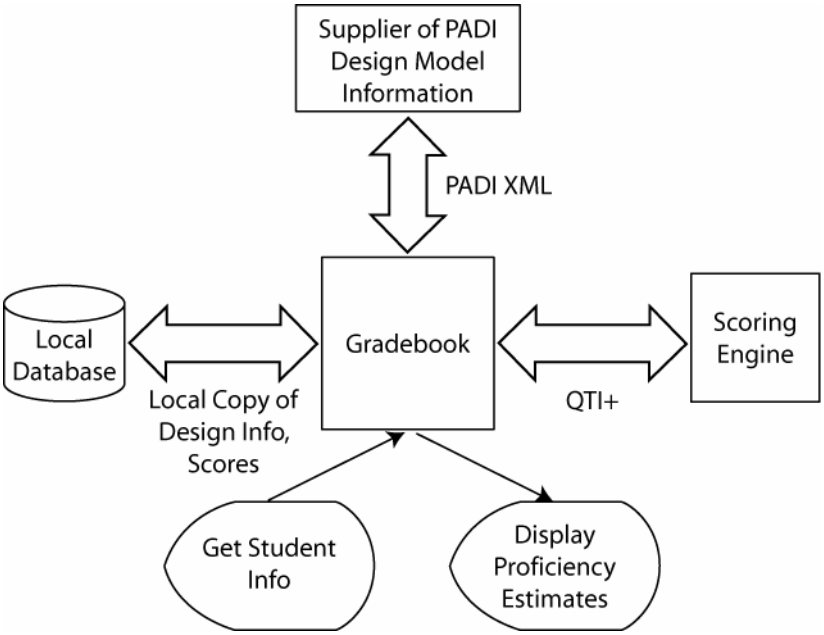
The second box within Figure 1 is labeled "Students' Item Scores" and represents instructors' scores for student performance (i.e., item scores, or, in more formal PADI terms, values of Observable Variables, for any number of students taking a given assessment). Gradebook makes no assumptions about assessment delivery or the evaluation of students' work that produced item scores. The mode of delivery could have been paper and pencil, computer-based, or oral; evaluations could have been automated or based on human judgment. In any of these cases, the resulting item scores can serve as input to Gradebook.

The third box in Figure 1 represents a request sent to the scoring engine (a separate system). Essentially, information from the PADI design system for assessments is combined with students' item scores and then sent to the scoring engine. The scoring engine produces estimates of student proficiency based on these Measurement Models and item scores.

Finally, the circle on the right of Figure 1 represents these proficiency estimates, which are returned from the scoring engine to Gradebook. Gradebook stores the estimates and presents a summary display suitable for an instructor. In other words, Gradebook does not itself

calculate estimations of proficiency, but relies on a service provider for this calculation. The interface with the scoring engine is defined in XML and uses the HTTP protocol, so Gradebook can interact with any provider of scoring calculation that implements the proper XML-over-HTTP interface. At the time of writing, all experimentation has been done with a single scoring engine, the BEAR Scoring Engine (Kennedy, 2005a).

Figure 2. Flow of Information in Gradebook



The representation of information flow in Figure 2 shows that Gradebook exchanges information with screens and external processes. The top of Figure 2 indicates a supplier of assessment design information in XML format (the PADI design system, in our experience). At the bottom of Figure 2, the two ovals indicate the display screens that instructors employ to retrieve and supply information. On the left, the cylinder shape represents the database used by Gradebook to store design XML and student scores. On the right, the box labeled "Scoring Engine" represents the separate process that Gradebook employs to update student proficiency estimates. The Scoring Engine communication channel uses the Question and Test Interoperability (QTI) XML format for student scores (IMS Global Learning Consortium, 2000), combined with other assessment design XML, hence the label "QTI+" used in the arrow connector between Gradebook and the Scoring Engine.

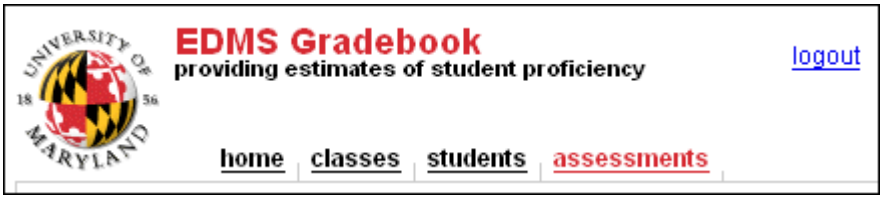
3.0 Implementation of Gradebook

In this section, we review the major aspects of the Gradebook implementation, starting with the Navigation bar and proceeding with descriptions of the major components of Gradebook: classes, students, and assessments. Additionally, Section 3.5 discusses scoring in more detail, and Section 3.6 describes the "bundling" of scores that can accommodate dependent items. Following this discussion of implementation, Section 4 discusses the lessons learned thus far with the existing Gradebook installation.

3.1 Navigation

Classes, students, and assessments are the core of the Gradebook system. The main links in the Gradebook header, as shown in Figure 3, will transport the instructor to the home page or to appropriate lists of classes, students, or assessments, respectively.

Figure 3. Navigation Header

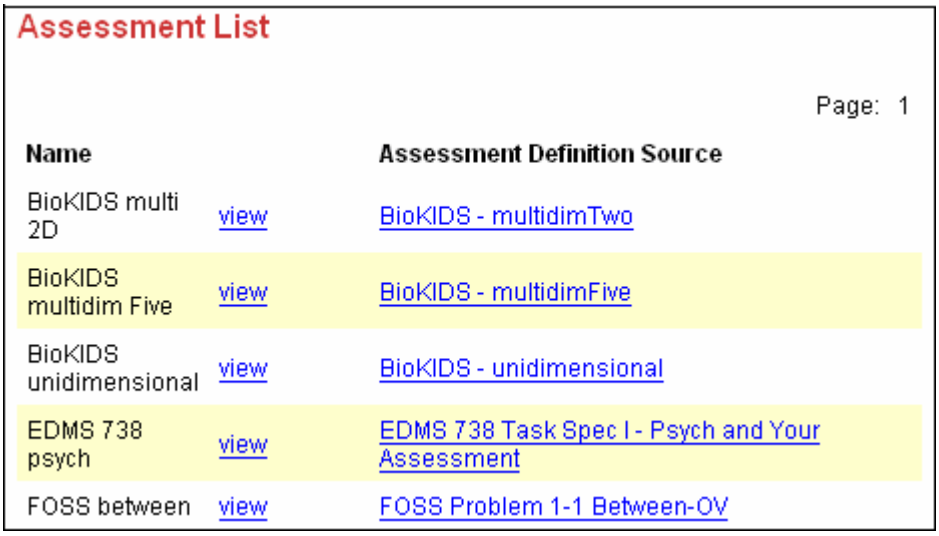


Robert Mislevy supervised the development of Gradebook and was the first customer; hence the Maryland logo and EDMS prefix in the header. There is also a login/logout link in the upper right corner, as shown in Figure 3. Logging in is optional, but only logged-in instructors have editing capabilities. Universal read-only access was deemed desirable for guest users of the prototype.

3.2 Assessments

An assessment in Gradebook is defined by a title, a URL, and some information about scoring methods. Any number of assessments can be added to Gradebook, and each of them will show up on the master list of all assessments. An assessment can be associated with any number of classes whose instructors choose to use the assessment. Associations between assessments and classes are discussed in Section 3.4.

Figure 4. List of All Assessments

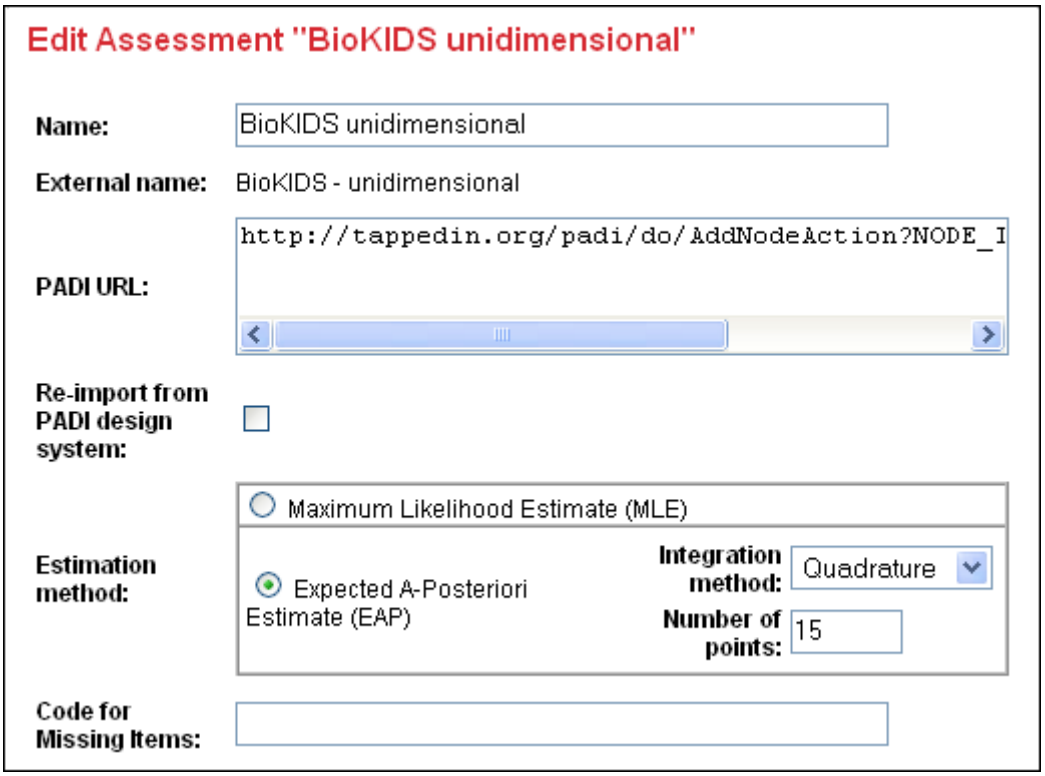


In Figure 4, five assessments are listed. For each assessment, instructors can view more information, edit the information (assuming they have sufficient permissions), delete the assessment, or see the full design information. Selecting the latter option accesses the assessment information within the original design supplier's database, following the URL specified by the instructor when the assessment was originally defined in Gradebook.

The names of the assessments in this example reflect some of the research goals and partners of the PADI project. There are three BioKIDS assessment variations, each employing a different Student Model with a different number of Student Model Variables (dimensions). BioKIDS is an implementation site for PADI (Songer et al., in press). Another partner is the University of Maryland EDMS department. Professor Mislevy kindly volunteered to describe his EDMS 738 seminar as a template in the PADI design system, which was subsequently used to test Gradebook importation of assessment design information. At the bottom of Figure 4, there is an assessment for FOSS, another implementation site (Long & Kennedy, in press).

The main work of designing an assessment is done in an external assessment-designing application, such as the PADI design system. Gradebook retrieves the task-design models from the design supplier in XML format by accessing a URL specified as part of the Gradebook assessment definition, as shown in Figure 5. Gradebook stores this XML file when the assessment is initially defined in Gradebook; subsequent changes made by the design supplier will not automatically impact Gradebook unless the instructor using Gradebook indicates that the task-design models should be updated. Updating can be accomplished by checking the Re-import box shown in Figure 5, but this is again a one-time change. This approach allows for the original assessment design to be changed without threatening ongoing usage of a consistent design by Gradebook.

Figure 5. Editing an Assessment



Gradebook communicates with the design supplier by means of simple HTTP requests and expects a well-defined XML document in return. Information flows from the design supplier to Gradebook in one direction only; Gradebook does not write any information to the design supplier.

3.2.1 Validation

Gradebook requires assessment design models (represented as XML documents) to exhibit certain qualities. These qualities are validated when the instructor indicates that an assessment should be downloaded from the design supplying application. The qualities for validation all concern the XML document's adherence to the structure of the PADI design models (see Riconscente, et al., 2005, for a full description of the PADI design models).

Gradebook downloads and validates the design model XML from a URL and assures that:

- The type of container object is a template or task specification.
- Exactly one Student Model contains all Student Model Variables (SMVs).
- All SMVs referenced by Measurement Models are defined within the Student Model.
- There is at least one Activity or at least one nested template or task specification (nested items are also validated, recursively).
- Any (optional) Measurement Models found contain exactly one Observable Variable (OV).

- Any (optional) intermediate OVs found have a proper reference to a final OV.

The XML is parsed into its constituent parts in order to validate this information, and the various parts of the design are stored separately within the database schema described in the Appendix.

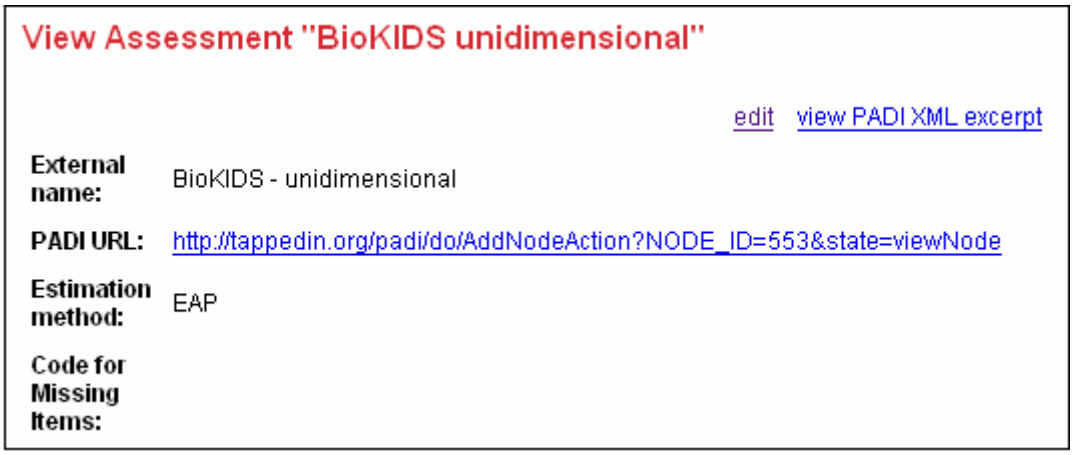
3.2.2 Scoring Configuration

In Figure 5 about editing an assessment, the attributes at the bottom of the screenshot concern how the assessment will be handled by the scoring engine. First, there is the "Estimation Method." The scoring engine allows two options for the estimation method: EAP and MLE, which are acronyms for "expected a posteriori" and "maximum likelihood estimation," respectively. See psychometric references such as Baker (2001), Kennedy (2005a), Lord (1980), Van der Linden and Hambleton (1996), and Wu, Adams, and Wilson (1998) for discussions of these statistical methods. Second, the entry for "Code for Missing Items" allows the specification of how missing data will be indicated. The scoring engine differentiates missing data from "0" scores. The default representation of a missing score is a blank field. However, it may be easier for humans to spot missing data if a specific code is used rather than just an omission (e.g., searching for "NO_ANSWER" is easier than trying to spot where some data are omitted entirely). For this reason, the scoring engine also is able to accommodate a specific code to designate missing data. The default value for this code is empty, to indicate that no special code should be used.

3.2.3 Viewing an Assessment and Its Associated XML

The assessment view page, shown in Figure 6, displays the attributes already discussed and offers some action links in the upper right corner.

Figure 6. Viewing an Assessment



The link in the upper right corner allows instructors to view some of the XML that will be sent to the scoring engine. A sample of that XML is shown in Figure 7.

Figure 7. Assessment XML Snippet

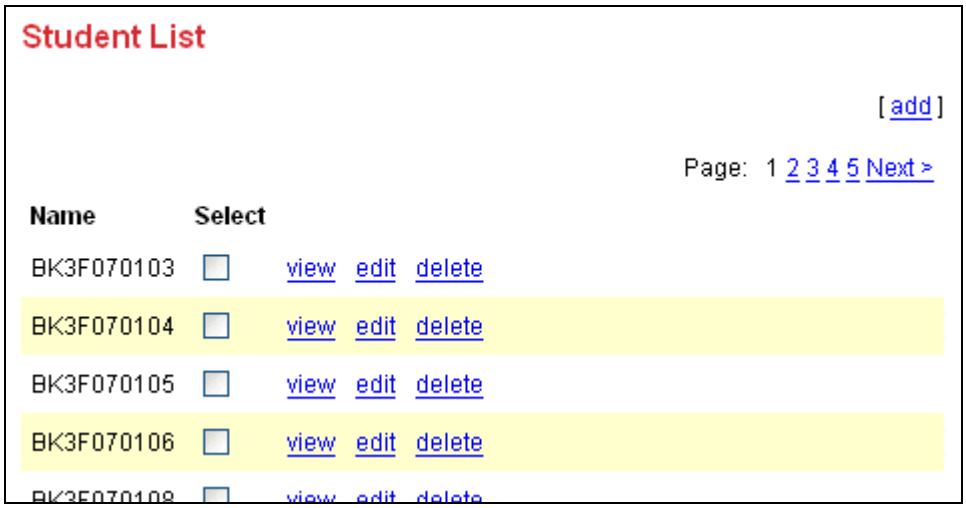
```
- <scoring_engine_input>
  - <scoring_engine_options>
    - <estimation_method>
        <MLE/>
      </estimation_method>
    </scoring_engine_options>
  - <STUDENT_MODEL_TYPE NODE_TITLE="Combined Inquiry and
    Content" NODE_TYPE_VERSION="3.40" ident="557">
    + <NODE_ANNOTATION></NODE_ANNOTATION>
    - <COVAR_MATRIX PART_LABEL="Covariance Matrix"
      ATTRIBUTE_ID="2761">
      - <COLUMN SM_VAR_ID="558" SM_VAR_NAME="Combined
        inquiry and content ability">
          <ROW SM_VAR_ID="558" SM_VAR_NAME="Combined
          inquiry and content ability" COVAR_VALUE="0.48105"/>
        </COLUMN>
      </COVAR_MATRIX>
    - <SM_DIST_MEANS PART_LABEL="Means Matrix"
```

In this snippet, the second through sixth lines have to do with options like the estimation method. In this instance, MLE (maximum likelihood estimation) is stipulated. After that, the Student Model element is specified, as provided by the design supplier. Although individual scores on each item for each student must be sent to the scoring engine, specifications for the Student Model and item parameters will be included just once in a request to the scoring engine, since these values are assumed to be the same for all of the students. In this way, the protocol is efficient, allowing unchanging design information to be transmitted once per request. More detail on the XML specification is given in the Appendix.

3.3 Students

In Gradebook, student records are simple data structures, including just a name, ID, and optional description. Students can be added individually via a Web form, or a group of students may be added at one time by importing a set of scores, as described in Section 3.5.

Figure 8. List of Students



In Figure 8, the students were entered with only an ID like "BK3F070103," so Gradebook treats the ID as the name of the student.

Adding an individual student is straightforward. Clicking on the "add" link in the Student List screen yields the entry screen shown in Figure 9.

Figure 9. Adding Students One at a Time via Web Form

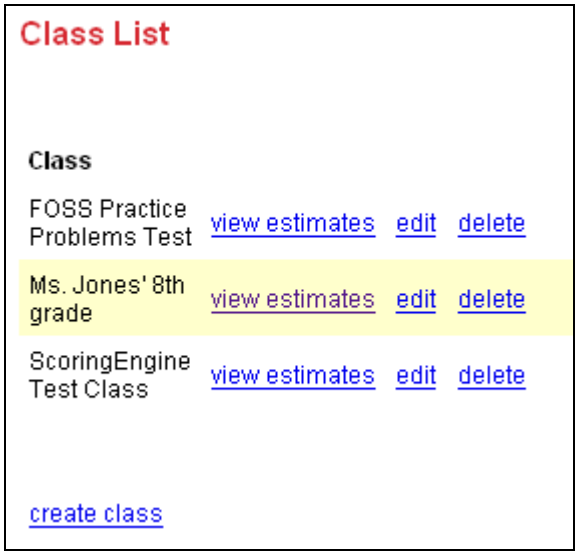
All students entered into Gradebook can be associated with any number of classes. A student identity need be entered only one time. Making associations between students and classes is described in the next section.

3.4 Classes

In Gradebook, classes are the data structures where students and assessments are associated. A student can be a member of multiple classes. An assessment also can be reused in different classes. It is assumed that all the students who are members of a class should be included in all

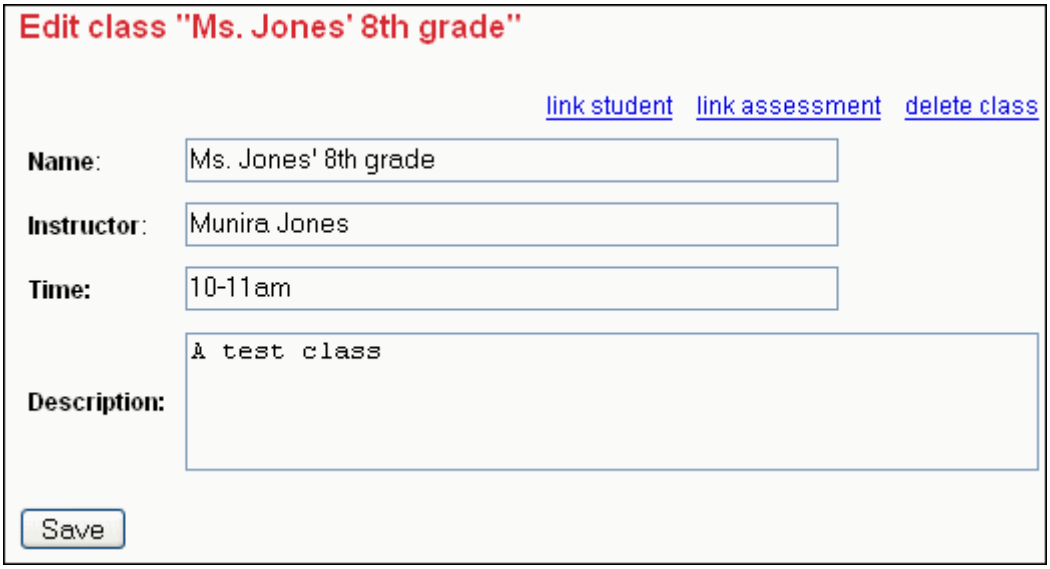
assessments associated with the class, although a student can have empty scores for a given assessment.

Figure 10. List of Classes



Three classes are shown in Figure 10, all examples that were created for research purposes. An instructor can create and edit classes by using the links shown on this page. Editing a class yields a page like that shown in Figure 11.

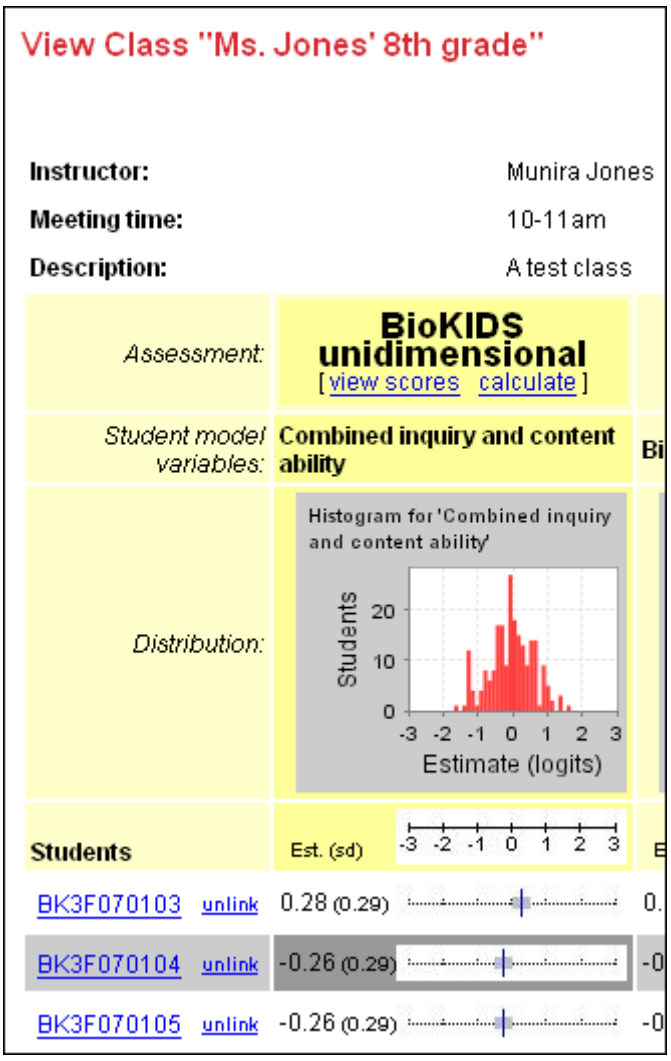
Figure 11. Editing Attributes of a Class



As shown in Figure 11, the definition of a class includes an instructor, a name of the class, and some other attributes. The important functionality of the class is in how it associates students with assessments, as provided by the links in the upper right corner of Figure 11.

Before we discuss the association procedures, consider the main display screen for a class, shown in Figure 12. For the main screen, Gradebook displays the proficiency estimates of all the students in the class. Gradebook could choose to just display scores as the main screen, but Gradebook assumes that instructors want to know the proficiency estimates that take into account all the responses of the students. In Figure 12, the histogram shows a summary of the class as a whole and has estimates for individual students at the bottom. In order for this kind of display to be generated, the instructor already has associated students with this assessment, provided scores for the students, and indicated that the scoring engine should calculate proficiencies. In other words, the main screen for viewing estimates is a result of many other steps, but it is also the view of most interest, assuming an instructor wants to see the proficiencies of students, not the individual item scores they received.

Figure 12. Viewing Proficiency Estimates of a Given Class



In the screenshot of Figure 12 (cropped to focus on just one assessment), the fictitious instructor Munira Jones has given an assessment referred to as "BioKIDS unidimensional" to her students who are indicated by IDs BK3F070103, BK3F070104, and so on, at the bottom.

A histogram summarizes the proficiencies in the class, graphing the range between -3 and 3. The units here are *logits*, a scale that is commonly used to report cognitive measurement data. The logit scale (technically, the "Log-Odds Unit" scale) is an equal interval scale. If the model holds, we can say that the difference in proficiency between a person with an estimate of 2 and a person with an estimate of 1 is the same as the difference between a person with an estimate of 3 and a person with an estimate of 2. In addition, the difference between 2 and 1 is twice as large as the difference between 3 and 2, with respect to suitably transformed[2] probabilities of correct responses to all the items in the domain. This differs from "percent correct" test scores, for which we cannot say that a person earning a score of 80% has twice as much knowledge as a person earning a score of 40%. The difference in the number of correct items for a person earning 80% compared with a person earning 40% is the same as the difference for a person earning 50% compared with a person earning 10%, in terms of percentage of correct responses. This does not necessarily mean that 80% correct represents twice as much knowledge or proficiency as 40% correct, nor that the difference between 80% and 40% represents the same difference in meaning or proficiency as the difference between 50% and 10%. Logits, on the other hand, are additive in the same way that inches are. A difference of 1 logit on a particular knowledge scale refers to the same difference in proficiency, regardless of which particular items a student may have completed, or which students were used to initially calibrate the scale.

Individual student estimates are available as decimal numbers in each student's row (see the bottom of Figure 12). Each student's proficiency estimate is also indicated as a tick on a line graph with a blue bar representing the standard deviation. For example, the first student, BK3F070103, has a proficiency estimate of 0.28, with a standard deviation of 0.29 on that estimate. The scale for the individual line graphs is the same as that used with the histogram.

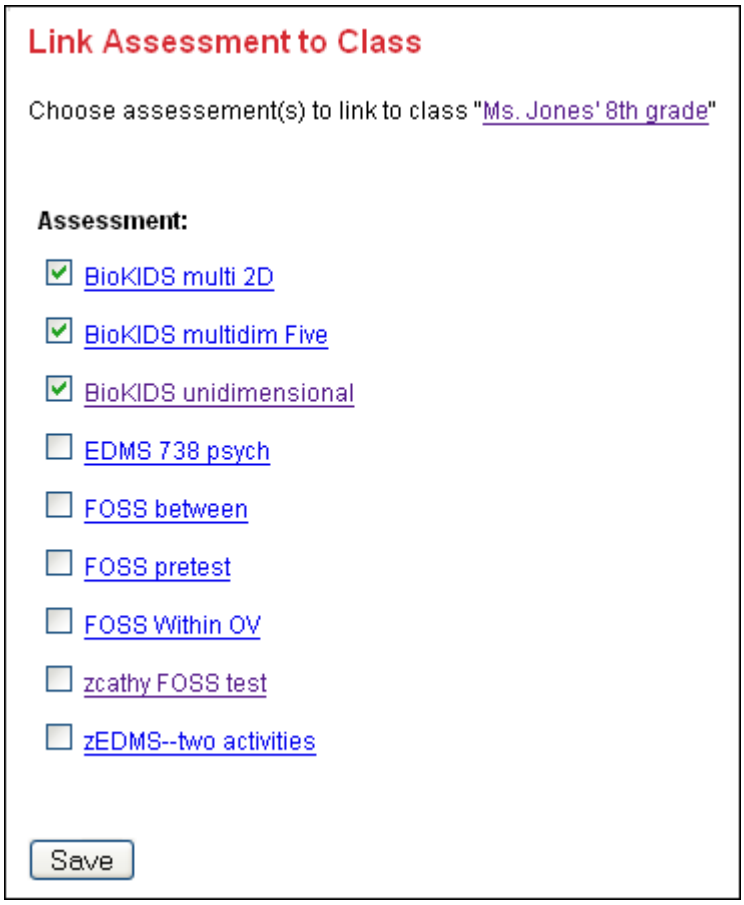
3.4.1 Associating Assessments

Figure 13 illustrates how assessments are associated with a class by checking the appropriate boxes.

Checking the box indicates that the assessment is associated with the target class. In Figure 13, three assessments are associated with Ms. Jones' class, while six other assessments are not. Associating an assessment with a class implies that all students in the class will take the assessment, although students may have empty scores for some or, occasionally, all items on a given assessment.

[2] Specifically, logit transformations: logit(p) = ln((p/(1-p)).

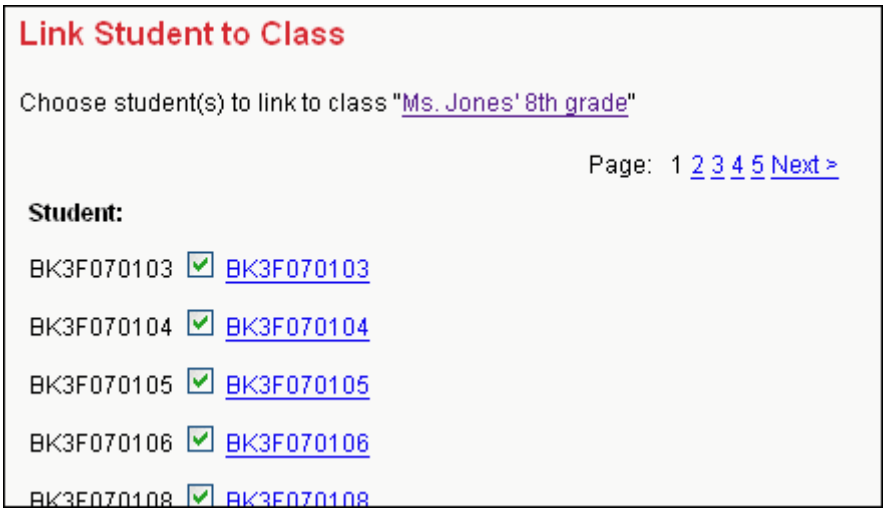
Figure 13. Linking Assessments to a Class



3.4.2 Associating Students

Students can be associated with classes in two ways: by an instructor's checking a box to associate individual students, as displayed in Figure 13, or in a more automated fashion, as described in Section 3.5.3 on importing scores.

Figure 14. Linking Students to Class



The top portion of the screenshot in Figure 14 shows several students associated with the class. In the subsequent pages, indicated by page links in the upper right of the figure, all students known to Gradebook are available for joining the class. A check box indicates whether or not each individual student is associated with the target class. A search function for particular students would be helpful here, rather than the multipage "browse" metaphor presented. However, the typical use case is to associate students with a class automatically during importation of scores, as described in Section 3.5.3.

3.5 Scores

The PADI design framework and design objects use the concept of Observable Variable to refer to an evaluated aspect of a student's performance. This is a generalization of the more familiar concept of an item score or, even more simply, a score. These are to be distinguished from test scores, commonly sums or percents-correct over several items. Gradebook calls the scoring engine to calculate IRT proficiency estimates to characterize students' performance across items.

3.5.1 Viewing Scores

Similar to the view of all the proficiency estimates of students in the class, Gradebook provides a view of all the scores for all students in the class for a given assessment, as shown in Figure 15.

Figure 15. Viewing Scores

View scores for assessment "BioKIDS unidimensional" within cla

	Observables:						
Students	BioKIDS pre/post item 1 edit scores	BioKIDS pre/post item 2 edit scores	BioKIDS pre/post item 3 edit scores	BioKIDS pre/post item bundled OV 4 edit scores	BioKIDS pre/post item bundled OV 5 edit scores	BioKIDS pre/post item bundled OV 6 edit scores	Bi pr ite ed sc
BK3F070103	1	1	1	3 (1, 1)	5 (1, 2)	5 (1, 1, 0, 1, 0)	
BK3F070104	1	1	0	2 (1, 0)	5 (1, 2)	4 (1, 0, 1, 0, 0)	
BK3F070105	1	0	1	4 (1, 2)	0 (0, 0)	7 (1, 1, 1, 0, 0)	
BK3F070106	1	1	1	4 (1, 2)	0 (0, 0)	8 (1, 1, 1, 1, 0)	
BK3F070108	1	1	1	2 (1, 0)	5 (1, 2)	5 (0, 1, 1, 1, 0)	
BK3F070109	0	0	1	3 (1, 1)	0 (0, 0)	4 (0, 1, 1, 0, 0)	

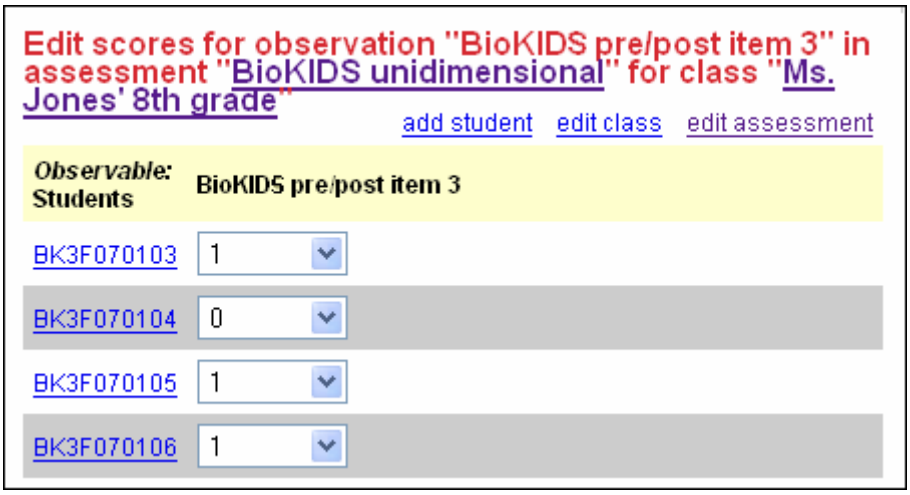
In this figure, each student has a row that displays the scores for individual items within the assessment, with each item in a separate column. For example, the first student, BK3F070103, has a score of 1 for the item "BioKIDS pre/post item 1" and also scores of 1 for the second and third items. The parenthetical numbers for the fourth, fifth, and sixth items have to do with item bundling, which is discussed in Section 3.6.

Not visible in Figure 15 are two links: one for exporting the scores as a text file and the other for viewing scores as XML. The XML format is discussed in the Appendix.

3.5.2 Score Entry via Web Forms

Instructors can enter or edit a score for a given student on a given item on a given assessment in a given class, as shown in Figure 16.

Figure 16. Editing Scores

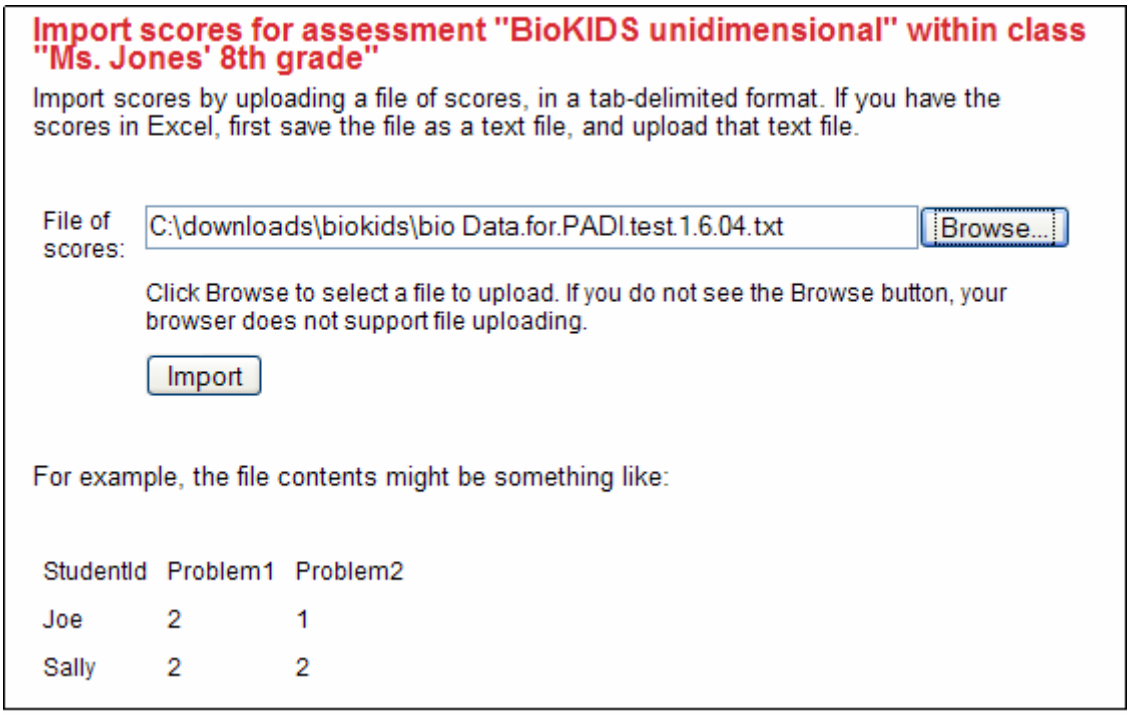


Score entry is limited to the range of possible answers via menu choice. In the item labeled "BioKIDS pre/post item 3" in Figure 16, three answers are possible: no response, 0 for incorrect, and 1 for correct.

3.5.3 Importing Scores

Instructors can import scores by uploading a tab-delimited file, as illustrated in Figure 17.

Figure 17. Specifying the Source of the Import [change extension to txt]



This feature facilitates the use of a local spreadsheet to enter student scores at the instructor's convenience, offline. The spreadsheet contents can be uploaded via a Web page. In order to validate the format of the incoming data, a confirmation screen is presented before the indicated file is saved into the database. This confirmation step allows Gradebook to offer several features to the instructor, including a means to reorder the data, create identification numbers for new students, or link existing students to the given class. In order to accomplish these multiple tasks on one screen, the confirmation page is relatively detailed. An explanation of the potential for data reordering is included at the top of the confirmation page, as shown in Figure 18.

Figure 18. Confirming Import (Part 1 of 2)

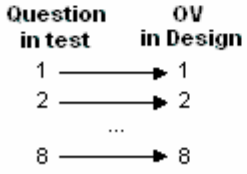
Confirm the format for importing scores for assessment "**BioKIDS unidimensional**" within class "**Ms. Jones' 8th grade**"

By default, the importer assumes that the order of the student responses found in the incoming file will match the order of the Observable Variables (OVs) found in the assessment design.

If anything is out of order, you can change the input file, or you can just manipulate the menus as necessary in "Matching Observable Variable" so that all items match properly.

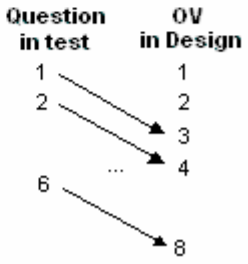
Example A (default match)

In a sample import file, say there are 25 students, each of which answered 8 questions (8 responses per student). There are 25 rows of data. Each row starts with a student ID and has 8 responses, so there are 9 columns in the the import file (ID + 8 responses). In that case, this confirmation page will show labels for responses 1 through 8 along the left-hand, vertical axis. The confirmation page will offer a menu of all of the possible OVs, in the order found in the design. Assume the design also has 8 OVs, in the same order as the responses in the import file. In that case, the instructor sees that the order of responses (columns) in the import file matches the order in the design, and the default matching is correct. The instructor needs to make no changes.



Example B (skipped items)

In another situation, say that the instructor has chosen to abbreviate the test of example A, skipping the first 2 items from the assessment design. There would be only 6 responses per student in the import file. In that case, this confirmation page will show labels for responses 1 through 6 along the left-hand, vertical axis. Since the design has 8 OVs, the OV menus will contain all of the 8 OVs, in the order found in the design. In this case, the match between import responses and OVs should be adjusted, as shown in the figure. The instructor indicates the proper matching by manipulating the OV menus such that the first response is matched with the third OV in the menu, the second response is matched with the forth OV in the menu, etc.



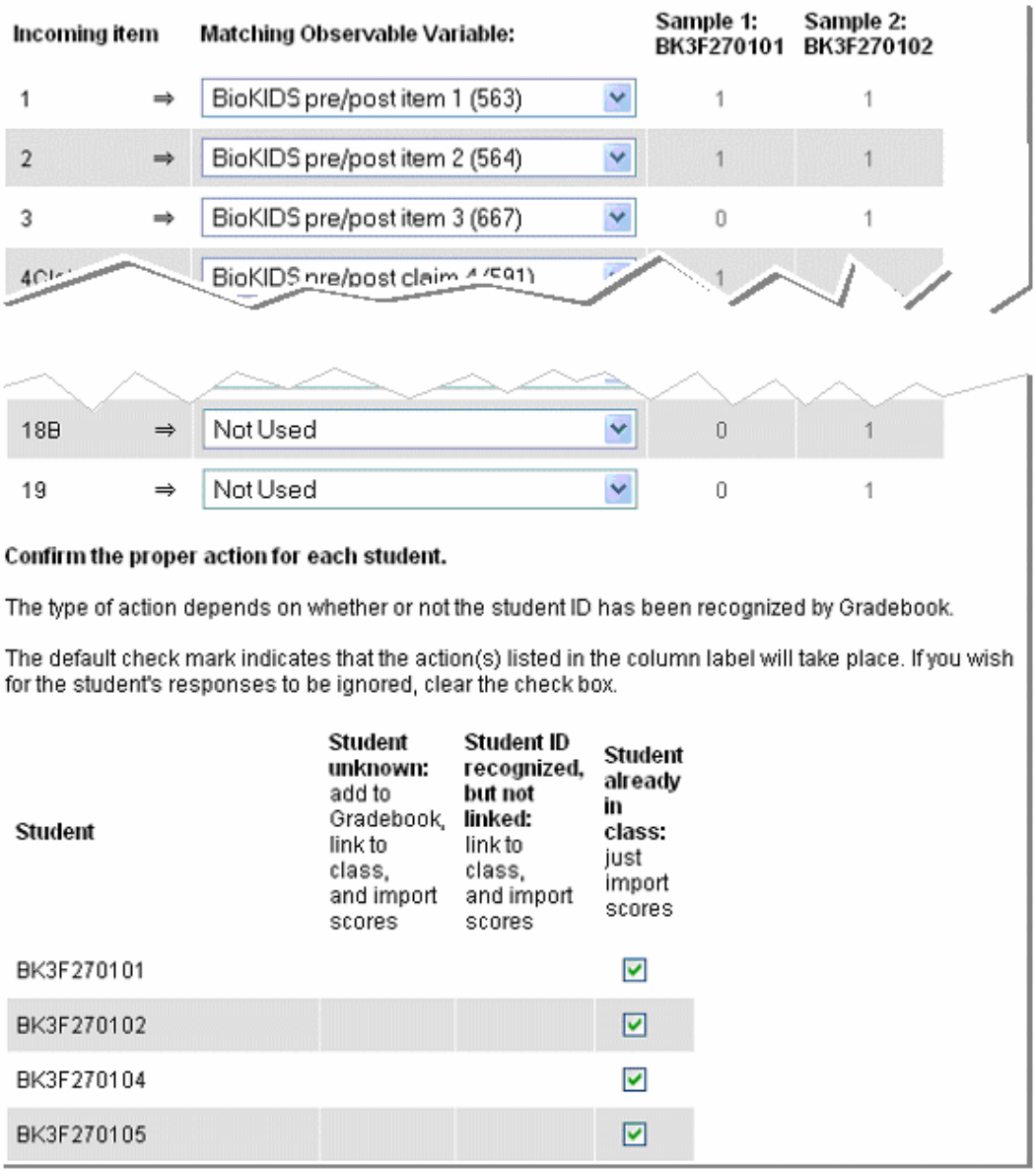
The menu of OVs include an option "Not Used" that will cause the importer to ignore the specified incoming question.

As explained in the text within Figure 18, the confirmation screen allows an instructor to change the order of Observable Variables (OVs) that are found in the assessment design models to match the incoming data. The default ordering may suffice in many cases.

The tutorial in Figure 18 is the top part of a detailed page that is further excerpted in Figure 19, which shows an abbreviated image with the OV mapping at top and student identification at bottom. As the explanation of Figure 18 states, the mapping of student responses to OVs is accomplished by manipulating, as necessary, a menu of OVs that includes all the OVs defined for the assessment design models. In Figure 19, the default mapping is correct. The import file has its responses in the order expected by the design. The label of the first response, "1," corresponds to the menu choice "BioKIDS pre/post item 1 (563)." The "(563)" in the OV title is the ID of the OV, appended by the system to the display label for the OV. In Figure 19, two columns of sample student scores are shown on the right for the first two students in the data set. In our sample data, the first two students have IDs BK3F270101 and BK3F270102, so their scores are shown in the right-hand columns as a sample of the actual responses found in the import file.

At the end of the OV-mapping menus, there are some unused data items, indicated by "Not Used." For example, response 19 in the imported data set is marked this way, indicating that there is no OV for this response. The importer will ignore response 19.

Figure 19. Confirming Import (Part 2 of 2)



Incoming item		Matching Observable Variable:	Sample 1: BK3F270101	Sample 2: BK3F270102
1	⇒	BioKIDS pre/post item 1 (563)	1	1
2	⇒	BioKIDS pre/post item 2 (564)	1	1
3	⇒	BioKIDS pre/post item 3 (667)	0	1
4Cl…		BioKIDS pre/post claim 4 (591)	1	
18B	⇒	Not Used	0	1
19	⇒	Not Used	0	1

Confirm the proper action for each student.

The type of action depends on whether or not the student ID has been recognized by Gradebook.

The default check mark indicates that the action(s) listed in the column label will take place. If you wish for the student's responses to be ignored, clear the check box.

Student	**Student unknown:** add to Gradebook, link to class, and import scores	**Student ID recognized, but not linked:** link to class, and import scores	**Student already in class:** just import scores
BK3F270101			☑
BK3F270102			☑
BK3F270104			☑
BK3F270105			☑

Furthermore, the import confirmation screen allows instructors to define students and class membership during the importation. The bottom of Figure 19 includes a list of student IDs with three columns of potential actions. In our sample situation, all of the student IDs in the import file already are known by Gradebook and already are linked to this class (e.g., from a previous assessment). Thus, only one of the columns, labeled "Student already in class," is populated with check boxes. The instructor typically follows the default here, using the data for all student IDs found in the import file.

In general, when Gradebook sees a student ID in an import file, there are three possible cases:

1. The student is unknown.
2. The student is known in Gradebook but is not a member of this class.
3. The student is already a member of this class.

For each of these cases, Gradebook can either use or ignore the student's data, so it offers a check box in the (only) appropriate column, corresponding to the correct case for the given student. Gradebook offers a checked box by default, assuming that data from all students should be used. Unchecking the box will cause Gradebook to ignore that particular student's scores.

In this example, student BK3F270101 is already known by Gradebook, so there is no need to show a check box in the left-most column for adding this student to Gradebook. Similarly, this student is already a recognized member of the class, so there is no need to offer a check box in the middle column to link this student to the class. Finally, there is a check box in the third column because it is appropriate to offer the instructor inclusion/exclusion choices about this student, who is already a member of the class. By leaving the "in class" box checked, the instructor indicates that the data for student BK3F270101 should be used to estimate the proficiency.

3.6 Item Bundling

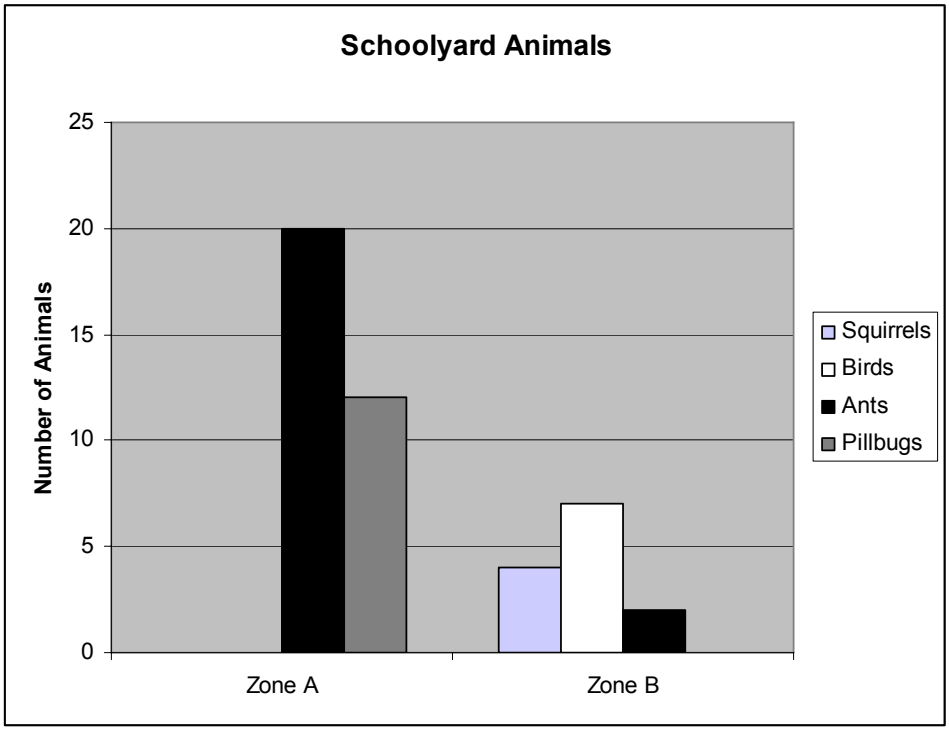
Importing scores becomes complicated when there are local dependencies among the items in an assessment. Dependencies arise when two or more student responses are somehow tied together: student performance on one item is not independent of performance on the other. This can occur, for example, when a single stimulus elicits multiple responses that are scored individually.

3.6.1 Example

For an example of conditional dependencies, consider the prompt for item 5 from BioKIDS, shown in Figure 20.

Figure 20. BioKIDS Item 5 Example

5. Using the graph below, predict which zone most likely has a tree in it and give one reason to support your prediction.



I think that zone _____ has a tree in it because

Note: For information on the BioKIDS project, see <http://www.biokids.umich.edu/>.

BioKIDS item 5 presents a graph and asks students to both fill in a blank for "zone" (with either "A" or "B" as an answer) and also to finish the sentence with a constructed response. BioKIDS terms the first part of the response the "claim" (identifying the zone) while the second part is termed the "evidence" (providing evidential reasoning with the word "because…").

The two responses are elicited from the same data and therefore are related to one another. For example, giving an incorrect claim is positively associated with also providing an incorrect explanation; the probability of giving a correct explanation is related to the probability of giving a correct claim. Thus, these two responses are conditionally dependent.

Several approaches to modeling conditional dependence appear in the psychometric literature. One approach that the BEAR Scoring Engine can accommodate is that of "bundling" conditionally dependent sets of items, and this technique also provides for associations more complicated than conditional independence (Wilson & Adams, 1995). From the perspective of the four-process architecture for assessment delivery, bundling means creating values of new observable variables from the patterns of scores across a set of conditionally dependent items. Thus, bundling is simply an additional step of item-level scoring.

3.6.2 Design Supplier Support

A design supplier can support response dependencies by indicating that the dependent responses should be bundled together during evaluation. Bundling makes a single, combined score out of multiple response scores. For example, in the BioKIDS item 5 example with two responses elicited by one stimulus, assume that the first response is judged right or wrong (1 or 0) while the second short essay response is rated on a scale from 0 to 2. Combining these two response scores implies that there will be two possibilities for the first response multiplied by three possibilities for the second response, yielding six possibilities. The bundled response score would scale from 0 to 5. In practice, however, the designer may eliminate some of these possibilities. For example, it is unlikely for a student to get a zero on the claim response, and then somehow get high marks for the explanation response. The designer may be able to eliminate (collapse) several of the six possible (combined) score categories after an analysis of student response patterns. This is a calibration exercise, using good judgment of how a given pattern of responses should be interpreted, noting in calibration data where few occurrences of a particular pattern might support collapsing one category into another.

A design supplier allows a designer to stipulate just how the permutations of a bundling evaluation should be handled—how the Output Observable Variable (also called the final OV) should be calculated from the Input Observable Variables (also called intermediate OVs). In Figure 21, an evaluation phase that does bundling is displayed.

Figure 21. Design System Screen: Translation Mapping for an Evaluation Phase That Bundles

Edit Action - Translation Chart for Bundling within Eval Phase
BioKIDS pre/post EP bundle 5

Select the proper category of the Output Observable Variable (OV) for each possible combination (permutation) of categories of the Input OVs:

BioKIDS pre/post claim 5	BioKIDS pre/post evidence 5	BioKIDS pre/post item bundled OV 5
0. incorrect	0. incomplete	0. claim 0, data 0
0. incorrect	1. partial	1. claim 0, data 1
0. incorrect	2. complete	2. claim 0, data 2
1. correct	0. incomplete	3. claim 1, data 0
1. correct	1. partial	4. claim 1, data 1
1. correct	2. complete	5. claim 1, data 2

Note: This is a screenshot from the PADI design system, not Gradebook.

On the right of Figure 21, an output (final) OV has six possibilities (0–5), and on the left are six permutations of intermediate OVs. This figure shows a one-to-one mapping. No categories were collapsed in this example. However, consider that getting the claim correct with no

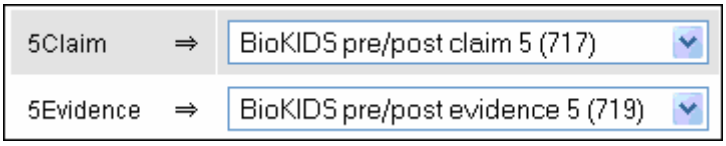
explanation could be construed by some designers as meaning the student was guessing, which would be equivalent to the first category of 0,0. Thus, there is an argument to collapse the intermediate OV combination described on the fourth row. The intermediate (1,0) category could be collapsed to the final OV category 0. In that case, Gradebook would not need final category 3, and the number of categories would be reduced from a total of six to a total of five.

3.6.3 Gradebook Support for Bundling

Gradebook receives the bundling information shown in Figure 21 as part of the design models of the assessment and understands that it should expect separate scores for separate responses from students. Gradebook knows it should perform the action indicated in the bundling evaluation phase by combining the separate responses into a single bundled score. Thus, Gradebook automatically conducts a simple mapping—a repetitive task at which computers excel. (An alternative design supply system could choose to hide the mapping from Gradebook, effectively turning off any evaluation actions in Gradebook.)

The import confirmation screen offers feedback about the bundling it has perceived, as shown in Figure 22, which is a snippet out of the same page shown in Figure 19.

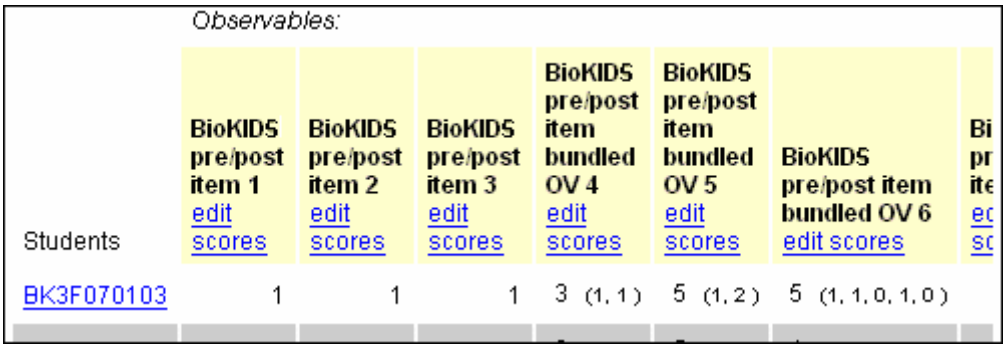
Figure 22. Confirming Import (Partial, Showing Bundled Responses for Item 5)



Here there are two parts to BioKIDS item 5. In the imported data file, the columns are labeled "5Claim" and "5Evidence," while the appropriate input OVs are shown in the menu choice. From the design models, Gradebook knows that item 5 requires bundling and shows the appropriate (intermediate) OVs to accept raw scores from the instructor.

After confirming the import format, instructors view the scores they imported, as shown in Figure 23.

Figure 23. Confirming Import (Viewing Scores) with Bundling Indicated by Parentheses



There are no intermediate OVs shown; only final OVs appear. For example, in the final OV labeled as "BioKIDS pre/post item bundled OV 5," Gradebook has combined intermediate inputs mentioned above into a single and final, bundled OV. In this OV 5 for the student BK3F070103, the student receives a score displayed as "5 (1, 2)," which is a way of saying that the bundled score is 5, as a result of getting a score of 1 on the first intermediate OV and 2 on the second intermediate OV. The parenthetical information is solely for reassuring instructors about how the mapping was done. The results passed on to the Scoring Engine contain only the single, bundled score.

3.7 QTI Format for Scores

To communicate with the scoring engine, Gradebook must assemble scores into a Question and Test Interoperability (QTI) XML document as specified by IMS Global Learning Consortium, Inc. (2000). The QTI specification provides a standard means to transport assessment responses from one learning-management system to another. By using a standard protocol, the designers of Gradebook intended to foster interoperability and open standards.

In addition to the QTI document for student responses, Gradebook also reformats some of the assessment design information it gets from the design supplier. QTI does not accommodate psychometric information like Student Models or Measurement Models, so this information is transmitted in a separate XML document specified by the PADI project, as described in the Appendix.

4.0 Lessons Learned and Future Directions

4.1 Visualization and Summarization

In early versions of Gradebook, estimates of student proficiencies were presented in a table of fractional numbers. For each student, Gradebook listed a proficiency estimate and a posterior standard deviation. These numbers are still part of the current display, but a simple line graph has been added to provide a different representation of the same information. Team members suggested aligning individual line graphs vertically for comparison purposes and presenting a summary of group performance in a histogram, so these features were added. In the future, usability studies could indicate the advantages and disadvantages of these displays versus others.

It would be easy to imagine graphical presentations of student performance over time. Like stock charts, such presentations could have all sorts of interesting statistical analyses. But a longitudinal presentation presupposes the measurement of the exact same Student Model Variables (proficiencies) using different assessments, as defined operationally by common items with the same item parameters across occasions. Gradebook is able to produce student estimates under these conditions, using the item parameters with which it is supplied. The responsibility for calibrating item parameters and maintaining a common scale across occasions is beyond the scope of Gradebook. A common situation for having identical Student Model Variables is to reuse an assessment as a pre- and posttest.[3] In this pre- and posttest situation, some graphical indication of student progress would be useful and could be something as simple as ornamentation (font color, typeface, etc.) of the student's name. As another improvement, we could keep track of whether students progress on different exams (i.e., progress or not) without calibrating the different exams against each other. In other words, we could provide graphical information about how often a student did better on a posttest compared with a pretest across various assessments (that included pre- and posttests) during the year.

Comparisons of groups who take the same assessment could be supported, either as subgroups within a class or as different classes. In the former case, it would be useful to allow the instructor to indicate contiguous zones within the range of a given Student Model Variable for a given assessment and given group of students. Each zone could be defined as a discrete proficiency level, and, for high-stakes situations, the zones could become the categories or grades for students. Among different classes, it might be useful to overlay histograms or otherwise provide comparison tools for graphical comparison between classes.

4.2 Navigation

The current navigation scheme is functional but could be improved. For students and assessments, the administration pages follow the simple pattern described by verbs like "view, edit, add, create, delete." In other words, viewing and editing information about a student's scores or an assessment are relatively straightforward processes.

[3] There are psychometric models that entertain the possibility of different student models at pretest and posttest, since differential patterns of learning may have introduced distinguishable dimensions in the data at the posttest that were not there at the pretest, or vice versa. See related work by Susan Embretson (Embretson, 1991).

Classes, however, are more complex in that they associate students with assessments. Classes also encapsulate the scores of students when they take an assessment, and they encapsulate estimates of proficiencies. The complexity starts showing up in navigation. In the list of classes, there is a link for editing a class that leads to an editing screen for a simple data structure that includes the class name and instructor's name. But unlike lists of assessments and students, the list of classes does not have a simple "view" link. If it did, would we expect a view of estimates or a view of scores? Assuming that an instructor will be interested in seeing estimates of student proficiency much more than seeing "raw" scores for students, we offer a link to "view estimates" for each class in the list of classes and no link for viewing all the item scores. So unlike more simple lists of students and assessments, the list of classes has a different result when clicking on "edit" versus "view." The "view estimates" link leads to a page that has estimates for all the assessments in the class, and from there, a link is provided to see the scores on which the estimates were based.

By presenting fewer choices, we avoid some complexity in the listing. For example, if a class has 20 assessments, we now provide just one link. If we were to present links for individual assessments, we would need 20 links (or 40 links if we included links to both the estimates and the raw scores) in a row for the class within the list of classes.

On the other hand, our current practice means, in this hypothetical case, presenting 20 histograms and the results for each individual student on a single page. That is too much information, and it would be slow to download and render. For more robust and generalized use, this assumption would need to be revisited. There may be a better way to provide navigation for classes, balancing the need for simplicity with the demands of a complex data set and Web page limitations.

Without a doubt, the major omission in navigation is the lack of a search function. Only browsing is provided currently. Again, a robust and generalized Gradebook would require functions to both search and browse.

4.3 Co-evolution with the PADI Design System and BEAR Scoring Engine

As described in Section 3.5.3, Gradebook benefited from features for bundling found within the PADI design system. This was a result of co-evolution of the two applications. When it became clear that bundling of dependent responses required the mechanical application of certain evaluation rules to a large set of data, and graduate students were spending hours transforming the data without any systematic check on accuracy, Gradebook was the obvious choice for automating the evaluation. However, Gradebook could only automate the evaluation if it knew the rules by which to evaluate student performance, which were not captured anywhere in a formal way. Adding the evaluation rules to the PADI design system was a significant task. In a similar way, Gradebook evolved with the scoring engine, establishing a communication protocol and a data dictionary as both systems were constructed.

4.4 Network Transfer

The protocol between Gradebook and the Scoring Engine is documented as part of the Scoring Engine (Kennedy, 2005a). Gradebook is only one of the clients of the Scoring Engine. Originally, a simple HTTP POST was used to transmit the two XML documents required by the Scoring Engine. However, data sets for large groups of students took several minutes to transmit. It

became clear that optimization was needed. So the scoring engine protocol was adapted to allow multipart forms that included compressed files. Compressing the XML files yielded over 95% reduction in size and a corresponding reduction in network transmission time.

Looking forward, the protocol should probably become a kind of Simple Object Access Protocol (SOAP) transaction in order to present a more standard protocol. Gradebook should also get some features for exporting and printing estimates and scores.

5.0 Conclusion

Gradebook is only a research prototype, with many shortcomings. However, it serves as a proof of concept that PADI assessment design models can lead to estimates of student proficiencies. Although a scoring engine is required for the core statistical calculations, Gradebook serves to simplify the conversion of model information and score information into the format required by the scoring engine. Furthermore, Gradebook can assist with minor evaluation tasks, and it presents proficiency estimates in a graphical format for easier comprehension by instructors.

References

Adams, R., Wilson, M. R., & Wang, W. C. (1997). The multidimensional random coefficients multinomial logit model. *Applied Psychological Measurement, 21,* 1–23.

Almond, R. G., Steinberg, L. S., & Mislevy, R. J. (2002). Enhancing the design and delivery of assessment systems: A four-process architecture. *Journal of Technology, Learning, and Assessment*, *1*(5). Available at <http://www.bc.edu/research/intasc/jtla/journal/v1n5.shtml>

Baker, F. (2001). *The basics of item response theory*. College Park, MD: ERIC Clearinghouse on Assessment and Evaluation, University of Maryland.

Embretson, S. E. (1991). A multidimensional latent trait model for measuring learning and change. *Psychometrika*, *56*, 495–516.

Hamel, L., & Schank, P. (2005). *Participatory, example-based data modeling in PADI* (PADI Technical Report 4). Menlo Park, CA: SRI International.

IMS Global Learning Consortium, Inc. (2000). *IMS Question & Test Interoperability specification: A review* (White Paper IMSWP-1 Version A). Burlington, MA: Author. Retrieved May 1, 2004, from <http://www.imsglobal.org/question/whitepaper.pdf>

Kennedy, C. A. (2005a). *Constructing PADI measurement models for the BEAR Scoring Engine* (PADI Technical Report 7). Menlo Park, CA: SRI International.

Kennedy, C. A. (2005b). *GradeMap v4.0 user guide* [Computer software manual]. Berkeley, CA: Berkeley Evaluation and Assessment Research Center, University of California. Retrieved August 30, 2005, from <http://bearcenter.berkeley.edu/GradeMap/docs/UserGuide4_0.pdf >

Long, K., & Kennedy, C. (in press). *Designing FOSS inquiry assessment tasks: An example of forward engineering using the PADI design system* (PADI Technical Report 19). Menlo Park, CA: SRI International.

Lord, F. M. (1980). *Application of item response theory to practical testing problems*. Hillsdale, NJ : Erlbaum.

Mislevy, R. J., & Riconscente, M. M. (2005). *Evidence-centered assessment design: Layers, structures, and terminology* (PADI Technical Report 9). Menlo Park, CA: SRI International.

Riconscente, M., Mislevy, R., Hamel, L., & PADI Research Group. (2005). *An introduction to PADI task templates* (PADI Technical Report 3). Menlo Park, CA: SRI International.

Songer, N. B., Gotwals, A. W., Bao, H., Haertel, G., Hamel, L., Kennedy, C., et al. (in press). *An illustration of PADI design capability in the BioKIDS project* (PADI Technical Report 13). Menlo Park, CA: SRI International.

Van der Linden, W. J., & Hambleton, R. K. (Eds.) (1996). *Handbook of modern item response theory*. New York: Springer.

Wilson, M., & Adams, R. J. (1995). Rasch models for item bundles. *Psychometrika, 60,* 181–198.

Wu, M. L., Adams, R. J., & Wilson, M. R. (1998). *ACER ConQuest: Generalized item response modelling software manual* [Computer software manual]. Melbourne, Australia: Australian Council for Educational Research, Ltd.

APPENDIX

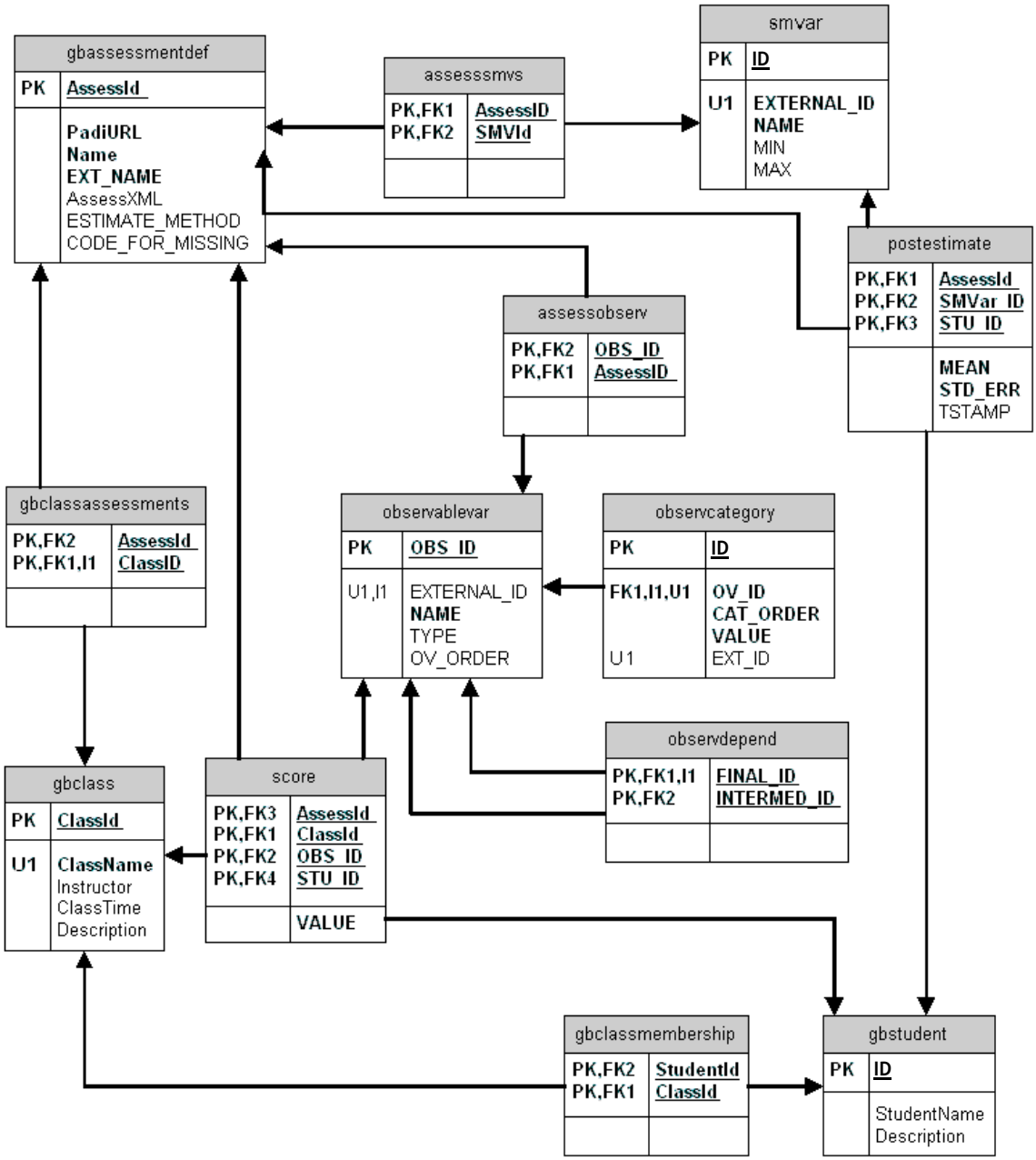
Technical Information

Appendix—Technical Information

A.1 Database Schema

Figure A-1 presents the database schema, which indicates how assessments, students, observables, and all the related details are stored.

Figure A-1. Database Schema



Classes (“gbclass,” lower left), students (“gbstudent,” lower right) and assessments (“gbassessmentdef,” upper left) are the primary entities with many foreign keys tied to them. Membership tables (gbclassmembership, gbclassassessments) associate students and assessments with classes.

When assessment designs are downloaded as XML files, those files are parsed for Student Model Variables (smvar) and Observable Variables (observablevar). Again, association tables (assesssmvs, assessobserv) make explicit the relations among assessments in their Student Model Variables and Observable Variables, respectively. Student scores (score) are duly recorded during manual scoring, as well as automatically during file import. After a request to the scoring engine, estimates of proficiency (postestimate) are stored for each student for each Student Model Variable.

In the middle of the schema, Observable Variables must be one of two types, intermediate or final. Intermediate Observable Variables are bundled together as described in Section 3.5.3 of the main text. Such intermediate observables are bundled to derive a score for a final observable, and the associations between intermediates and finals are duly recorded (observdepend) when the assessment design is parsed.

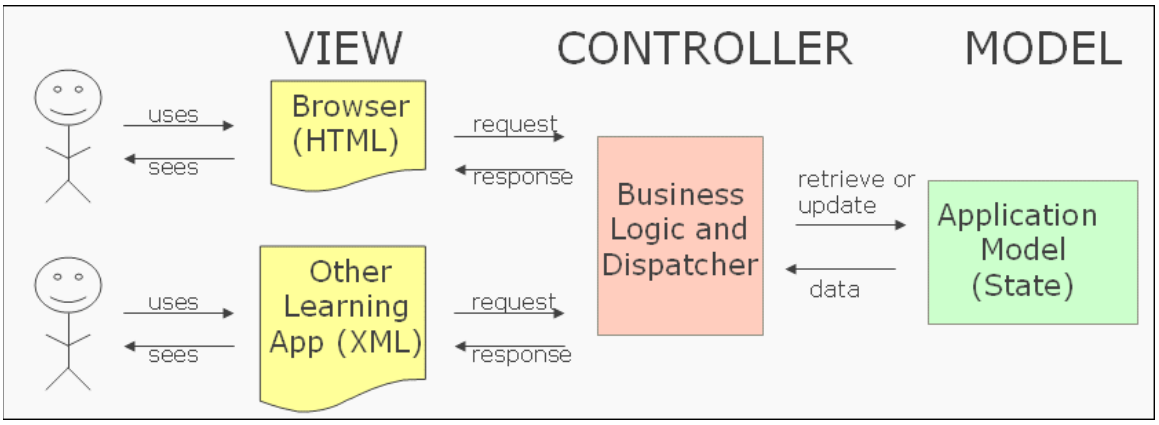
Finally, most observables have categories that are stored (observcategory) in order to provide a menu for manual input and also validate imported scores.

A.2 Request Flow

To show the flow of requests, a flow diagram and table are pictured in Figure A-2. A Unified Modeling Language (UML) diagram would be largely uninformative because Gradebook is based on a Web application framework called Expresso (available at <http://www.jcorporate.com/econtent/Content.do?state=template&template=2&resource=636>), wherein data model objects are directly related to the database entities shown in the database schema. In other words, the UML for data model objects would be redundant, given the entity definitions in the schema.

Furthermore, the flow of requests through the Controller subclasses of Expresso is better demonstrated as a simple, finite state machine. Expresso is based on a model-view-controller design with dispatching to states determined by URL parameters.

Figure A-2. Model-View-Controller Design



In Expresso, the view layer is accomplished with JavaServer Pages (JSP) technology, and the model is accomplished with an object-relational layer called DBObjects that communicates with the database.

The dispatching and control logic is accomplished with Java servlet technology as implemented in subclasses of the Controller class in Expresso. The Expresso framework implements a finite state machine where one state in the machine is fully described by a Controller (the file part of a URL) and an HTTP parameter called "state." The underlying Java implementation maps the state parameter to a method name in the specified Controller class, so the URL specifies both the class and the method for dispatching.

The dispatch table and view mapping, as realized in Gradebook, are shown in Table A-1.

Table A-1. Requests and Rendering Map

URL Base, Controller Class	State Name	Rendering JSP
/Welcome		
com.codeguild .umdgradebook. controller.Welcome		
	prompt	/welcome.jsp
/Classes		
com.codeguild .umdgradebook.controller .Classes		
	list	/classes/listclasses.jsp
	promptCreate	/classes/promptCreate.jsp
	promptDelete	/classes/promptDelete.jsp
	showClass	/classes/showclass.jsp
	promptEditClass	/classes/promptEdit.jsp
	promptAddAssessment	/classes/promptLinkAssess.jsp
	promptAddStudent	/classes/promptLinkStudent.jsp
/GradebookLogin		
com.codeguild .umdgradebook.controller .GradebookLogin		
	promptLogin	/security/login.jsp
	processLogin	/welcome.jsp
	processLogout	/security/logout.jsp
	promptChangePassword	/expresso/jsp/register/change.jsp
	processChangePassword	/expresso/jsp/register/status.jsp
	emailValidate	/expresso/jsp/register/status.jsp
	promptSendPassword	/expresso/jsp/register/sendPassword.jsp
	processSendPassword	/expresso/jsp/register/status.jsp

Table A-1. Requests and Rendering Map (continued)

URL Base, Controller Class	State Name	Rendering JSP
/Students		
com.codeguild .umdgradebook.controller .StudentController		
	promptCreateStudent	/students/promptCreate.jsp
	viewStudent	/students/view.jsp
	promptEditStudent	/students/promptEdit.jsp
	promptDeleteStudent	/students/promptDelete.jsp
	list	/students/list.jsp
/Assess		
com.codeguild .umdgradebook.controller .Assessments		
	listAssessments	/assessments/list.jsp
	viewAssessment	/assessments/view.jsp
	promptCreate	/assessments/promptCreate.jsp
	promptEditAssessment	/assessments/promptEdit.jsp
	promptDelete	/assessments/promptDelete.jsp
	promptScoreObs	/assessments/promptScoreObs.jsp
	viewScores	/assessments/viewScores.jsp
	promptImport	/assessments/promptImport.jsp
	confirmImport	/assessments/confirmImport.jsp
	exportScores	/assessments/exportScores.jsp

A.3 XML Specification

Two XML documents are required to make a request of the scoring engine. First, the student responses must be formatted in the QTI 1.2 language of XML, as described on the IMS Global site at <http://www.imsglobal.org/question/qtiv1p2/imsqti_res_infov1p2.html> and in associated documents on that site. Second, the PADI psychometric information about Student Models, Measurement Models, and so on, must be formatted in a proprietary XML format (QTI does not accommodate this information). The precise XML schemas for the various parts of the proprietary format for the scoring engine are described on the BEAR site at <http://bearcenter.berkeley.edu/padi/schemas/> and also in BEAR literature about the scoring engine. Both the QTI format and the PADI psychometric XML format are summarized below. For more specific information, consult the QTI documentation and BEAR documentation, respectively.

A.3.1 QTI

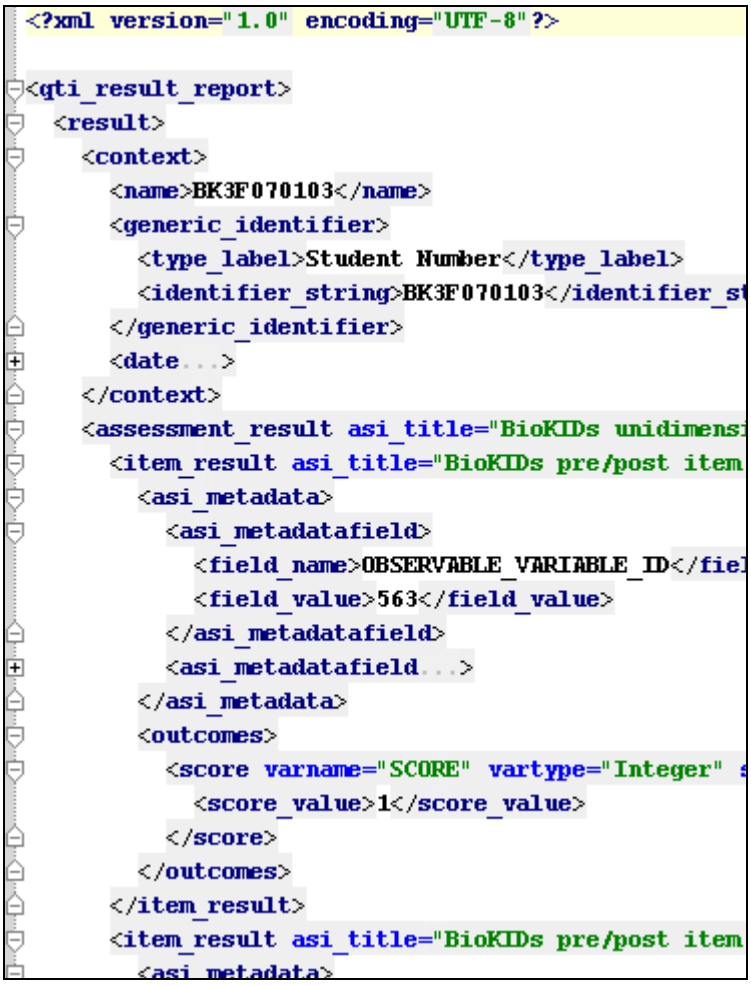
The main hierarchy of concern in our case is described in Table A-2, where the left column has indentations to imply hierarchical nesting and the description on the right explains the information found within the tag.

Table A-2. Annotated Summary of Tags within QTI

Tag	Comment
<qti_result_report>	Root tag: all students
<result>	One student
<context>	All identification for this student
<assessment_result>	One assessment for this student
<item_result>	One item within assessment
<asi_metadata>	All meta-data about this item
<asi_metadatafield>	One piece of metadata (e.g., ID of item)
<outcomes>	All outcomes for this item
<score>	One score for this item

A sample of such QTI language XML can be found in Figure A-3.

Figure A-3. Sample of QTI



In this example, the first student is identified with ID BK3F070103 inside the tag named "identifier_string;" this ID also happens to be the name used for the student. Also, the ID for the observable variable is "563" inside the tag "field_value," which is embedded as metadata for the appropriate item. For that item, the student received a score of "1" inside the tag "score_value."

A.3.2 Psychometric XML Plus Scoring Engine Options

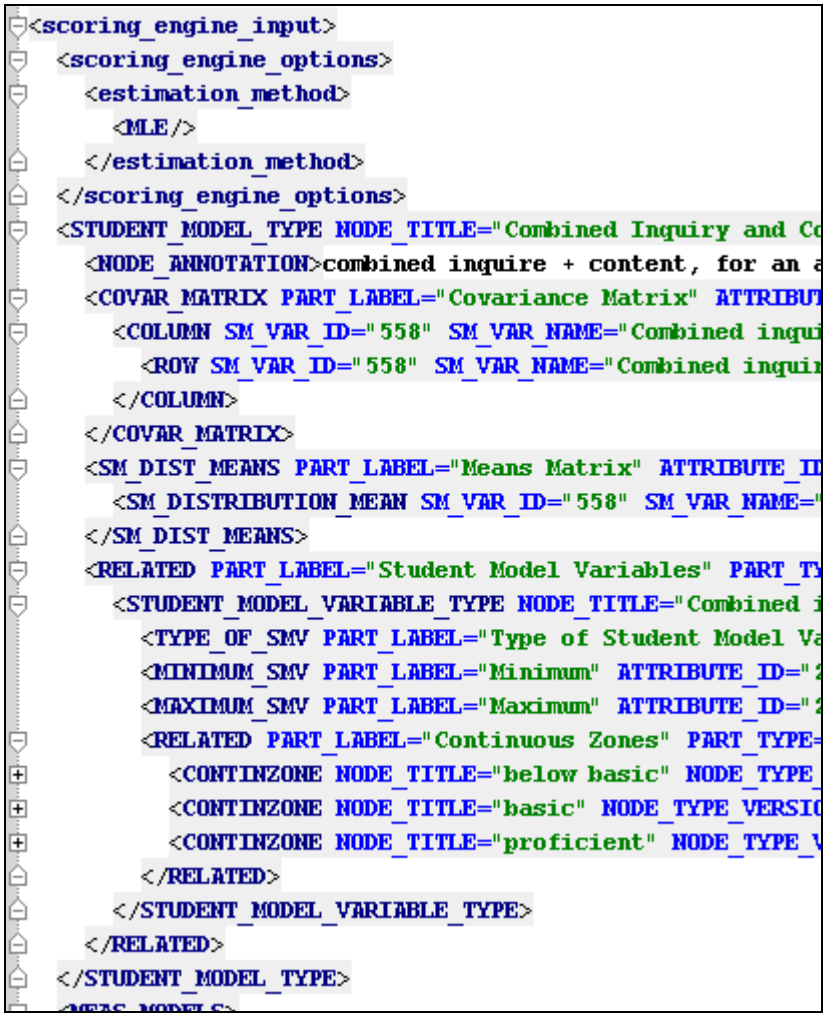
The main hierarchy of concern for the psychometric XML is described in Table A-3, where the left column has indentations to imply hierarchical nesting and the right-hand column description explains the information found within the tag.

Table A-3. Annotated Summary of Tags within Psychometric XML

Tag	Comment
<scoring_engine_input >	Root tag: all psychometric info and scoring options
<scoring_engine_options>	All scoring options
<STUDENT_MODEL_TYPE>	Student Model (only one allowed)
<MEAS_MODELS>	All Measurement Models
<MEASUREMENT_MODEL_TYPE >	One Measurement Model

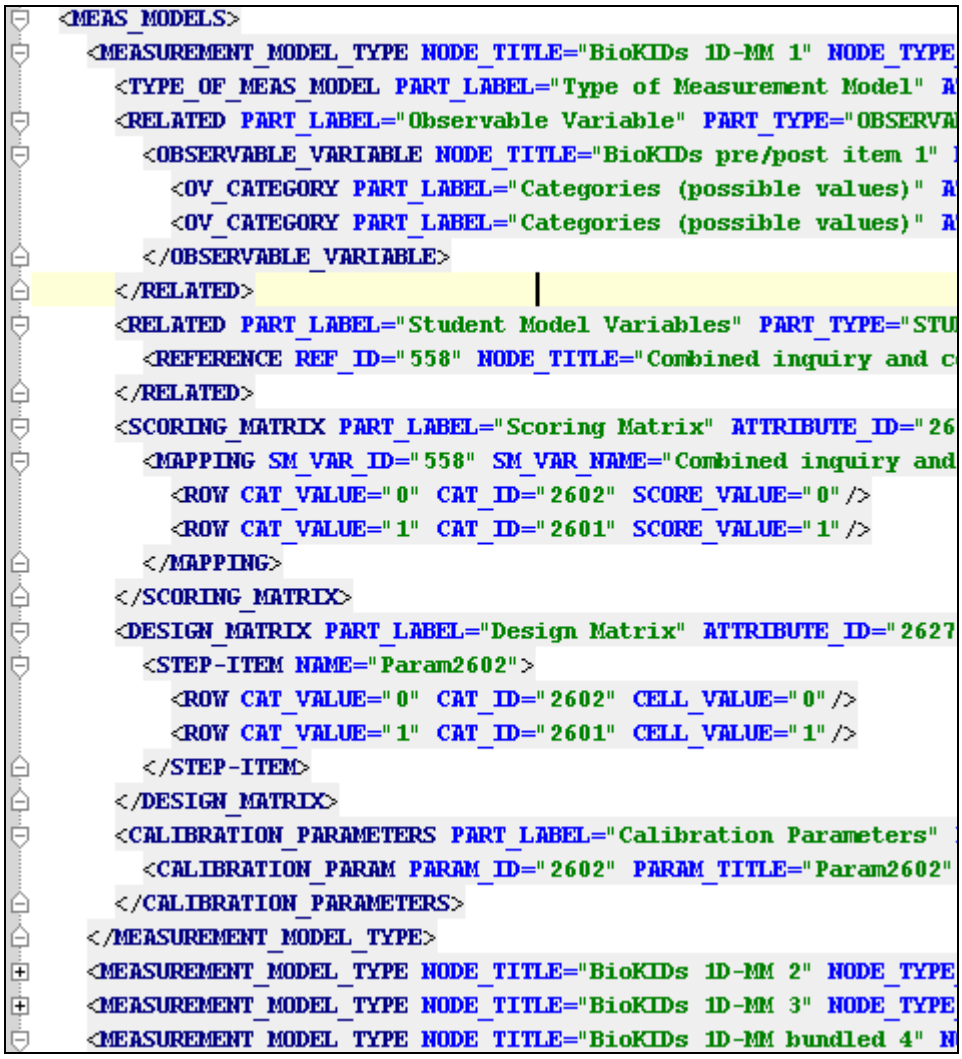
A sample of such psychometric XML can be found in Figure A-4.

Figure A-4. Sample of PADI Psychometric XML



In this example, the scoring engine is instructed to use the MLE (maximum likelihood estimation) method, and the single Student Model has the title "Combined Inquiry and Content." No Measurement Model is visible in Figure A-3. A continuation of the XML document is shown in Figure A-5.

Figure A-5. Continuation of PADI Psychometric XML Showing Measurement Model



As indicated, the first Measurement Model has title "BioKIDS 1D-MM 1" and has a reference to a single Student Model Variable. A reference like this implies that the object referred to has already been fully described previously in the document. The Measurement Model also has an Observable Variable, a Scoring Matrix, a Design Matrix, and Calibration Parameters. Several other Measurement Models follow the first one, but in this display, they have been collapsed (the contents within the tag hidden from view) at the bottom of the figure. For more detail about XML structure for psychometric information, consult the technical reports available on the PADI Web site, <http://padi.sri.com>.