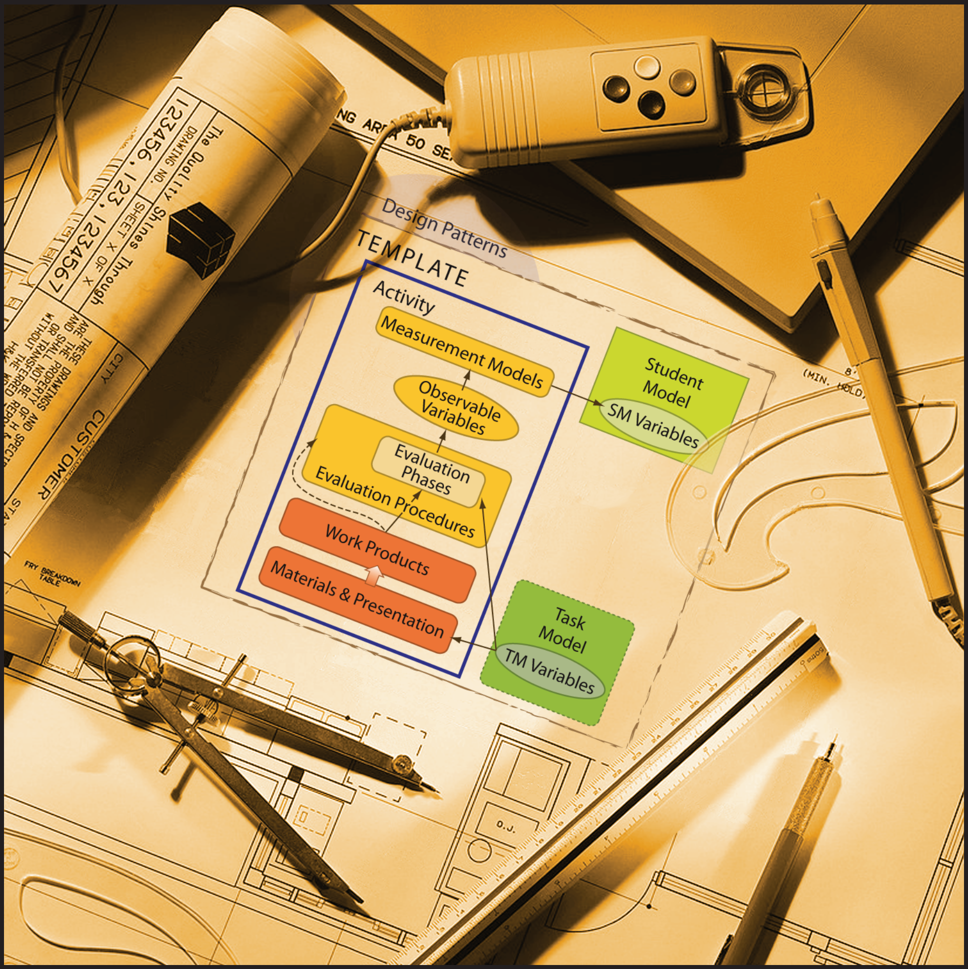


PADI Technical Report 11 | January 2006

A Wizard for PADI Assessment Design

PADI | Principled Assessment Designs for Inquiry

Lawrence Hamel, CodeGuild, Inc.
Patricia Schank, SRI International

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

PRINCIPLED ASSESSMENT DESIGNS FOR INQUIRY
TECHNICAL REPORT 11

A Wizard for PADI Assessment Design

Prepared by:
Lawrence Hamel, CodeGuild, Inc.
Patricia Schank, SRI International

Acknowledgments
This material is based on work supported by the National Science Foundation under grant REC-0129331 (PADI Implementation Grant).

CONTENTS

FIGURES

TABLES

Abstract

The Principled Assessment Designs for Inquiry (PADI) project (<http://padi.sri.com>) includes a design system that provides a structure for assessment designs, intended to support and encourage assessment designs with clear rationales. This PADI *structure* can be summarized as a *template* of an assessment design with many parts. Designing an assessment becomes, in essence, filling in a *template* intelligently and making all the choices and interconnections among the various parts of the *template*.

For an analogy to a *template*, consider a U.S. federal income tax form, which must be filled out by understanding the interconnected rules and assorted constraints. To mediate the complexity, popular tax software provides an interview format, where filling out the form is reduced to answering a series of questions.

Likewise, the PADI design system has a means to create and conduct interviews, or wizards, which prompt for decisions about assessment design. In a first prototype, we implemented a wizard that prompts for some selection criteria, eventually matching the answers supplied by the interviewee to an existing *template* that already has a substantial amount of information entered. This matching *template* is then duplicated and can be customized further by the assessment designer who uses the system.

This report includes an overview of the purpose and implementation of the wizard system, along with brief discussions of the initial impressions by assessment designers who have used it.

1.0 Background

1.1 PADI

The Principled Assessment Designs for Inquiry (PADI) project (<http://padi.sri.com>) is led by Principal Investigators Geneva Haertel of SRI International and Robert Mislevy of the University of Maryland. The PADI project aims to provide a practical, theory-based approach to developing quality assessments of science inquiry by combining developments in cognitive psychology and research on science inquiry with advances in measurement theory and technology. One of the outcomes of the project is an articulation of a conceptual framework for designing complex assessment tasks. The conceptual framework defines a system of interrelated objects that contain assessment information, including a central *template* object that represents a blueprint for an assessment. Designing an assessment becomes, in essence, filling in a *template* intelligently, and making all the choices and interconnections among the various parts of the *template*. The PADI project includes a design system that provides a Web-based editor for assessment designs, as described by *templates*. A *template* has 23 separate attributes, many of which are relations to other complex objects, each with its own set of attributes. It can be daunting to complete a *template* from scratch. For more details on the role and structure of *task templates* in PADI, see Riconscente, Mislevy, Hamel, and PADI Research Group (2005).

As a very brief introduction to *templates*, see the list and definitions of *template* attributes in Table 1. *Templates* are blueprints for assessment tasks that combine task environment information with evidence evaluation logic. *Templates* are also known as task-evidence shells. *Templates* can vary from abstract, general ideas to concrete specifications, ready to generate assessments. A *template* generally retains some flexibility, such as Task Model Variables that have not been specified yet. When every variable in a *template* is decided and specified for a particular assessment, the *template* becomes a *task specification*, something that is ready for use in generating assessments.

Table 1. Template Attributes

Attribute	Description
Type	Indicates whether the object is a finished, complete, concrete task specification or a template, which is abstract and general.
Student Model Summary	Outlines the student models in the template.
Student Models	Associations with (potentially shared) objects of type: Student Model.
Measurement Model Summary	Describes the nature and requirements for Measurement Models used in this template. For example, one could mention whether the template requires a multidimensional model, or whether items have dependencies.
Evaluation Procedures Summary	Outlines general requirements for Evaluation Procedures.
Work Product Summary	Outlines the things created by the student.
Task Model Variable Summary	Outlines all the Task Model Variables that are used by this template.
Template-level Task Model Variables	Associations with (potentially shared) objects of type: Task Model Variable.

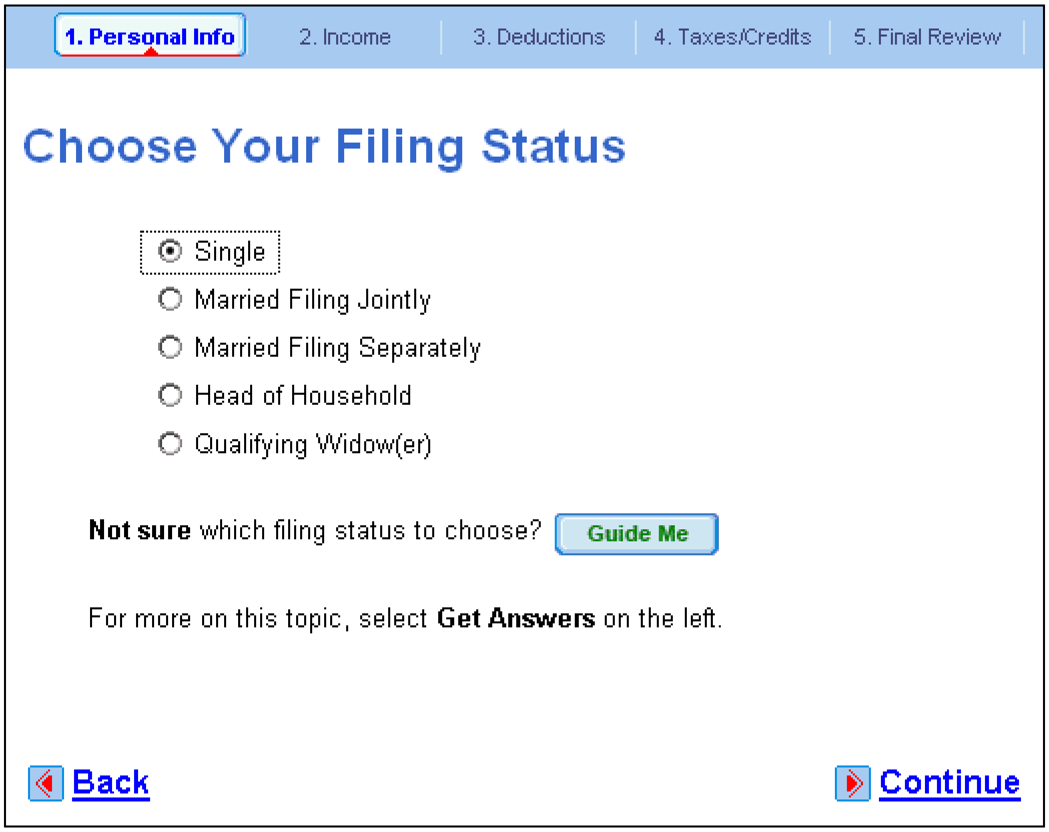
Table 1. Template Attributes *(continued)*

Attribute	Description
Task Model Variable Settings	The exact choices made from among those allowed for each Task Model Variable (TMV). In other words, the designer has specified a given Task Model Variable, and it is no longer variable. The template is "pinned" to use this setting. Settings apply to the template/TMV combination. The same TMV may have different settings in different templates if it is associated with more than one template. Templates may also have associated Activities, and these Activities may have associated TMVs, but any setting for an "Activity" TMV is still controlled by the template. Settings apply to the template, not to individual Activities, even though a TMV may show up under the Activity only.
Materials and Presentation Requirements	Specifies how the stimuli are presented to the student and any large-scale needs, like having a large room.
Template-level Materials and Presentation	Associations with (potentially shared) objects of type: Materials and Presentation.
Materials and Presentation Settings	The exact choices made from among those allowed for each Materials and Presentation (M&P) item. In other words, the designer has specified a given Materials and Presentation choice, and it is no longer variable. The template is "pinned" to use this setting. Settings apply to the template/M&P combination. The same M&P may have different settings in different templates if it is associated with more than one template. Templates may also have associated Activities, and these Activities may have associated M&Ps, but any setting for an "Activity" M&P is still controlled by the template. Settings apply to the template, not to individual Activities, even though an M&P may show up under the Activity only.
Activities Summary	An overview of all the Activities included.
Activities	Associations with (potentially shared) objects of type: Activity.
Tools for Examinee	Things provided to or permitted for use by the examinee.
Exemplars	Associations with (potentially shared) objects of type: Task Exemplar.
Educational Standards	Associations with (potentially shared) objects of type: Educational Standard.
Design Patterns	Associations with (potentially shared) objects of type: design pattern.
I am a kind of	Associations with other objects that are more abstract or more general than this object. For example, a dog is a specific kind of animal.
These are kinds of me	Associations with other objects that are more concrete or more specialized than this object. For example, animal is a general category that includes specific kinds of dogs.
These are parts of me	Associations with other objects that contain or subsume this one. For example, a windshield is a part of an automobile.
Online Resources	Relevant items that can be found online (URLs).
References	Notes about relevant items, such as academic articles.

1.2 TurboTax

For an analogy to a *template,* consider a U.S. federal income tax form, which must be filled out by understanding the interconnected rules and assorted constraints. To mediate the complexity, popular tax software like TurboTax by Intuit, Inc. (<http://intuit.com>) provides an interview format, where filling out the form is reduced to answering a series of questions (see Figure 1).

Figure 1. TurboTax Screen[1]



Choose Your Filing Status

- (●) Single
- ( ) Married Filing Jointly
- ( ) Married Filing Separately
- ( ) Head of Household
- ( ) Qualifying Widow(er)

Not sure which filing status to choose?

For more on this topic, select **Get Answers** on the left.

[1] *From TurboTax 2005 by Intuit, Inc. Copyright 2005 by Intuit, Inc. Reprinted with permission.*

TurboTax presents a question and answer format, along with navigation possibilities to go to the previous question (the Back button) or random access to anywhere in the interview via a menu at top (1. Personal Info, 2. Income, and so on). TurboTax attempts to isolate one issue at a time, simplifying the focus and offering additional help on separate screens accessible by the navigation button Get Answers and a Help menu. Likewise, the PADI design system has a means to create and conduct interviews, also known as wizards, which prompt for decisions about assessment design.

1.3 Wizards in the Literature

Wickham, Mayhem, Stoll, Touley, and Rouiller (2002) describe a rationale and method for creating wizards, including the following general goals of wizard design (p. 89):

- Hide the complexity of the underlying task.
- Ask the users simple questions.
- Include as many defaults as possible so users do not need to enter all information for task completion.
- Reduce the number of entry fields by replacing them with choices in list boxes or radio buttons.
- Maintain task cohesiveness.
- Provide a clear indication of mandatory fields.
- Keep the user in control.

They also discuss the possibility of designing wizards for novice and expert audiences together. They suggest integrating expert and novice functions within an integrated wizard by providing optional controls, optional pages, and dialogs that proceed from a button labeled "advanced."

Tidwell and Fuccella (1997) describe a system that uses Standard Generalized Markup Language (SGML) to provide a flexible wizard creation system called TaskGuides. It also has a "meta-wizard"—an interview process that can create an interview process. They offer a syntax that permits branching and other complex behavior.

2.0 The Design Process

In this section, we lay out our design goals and describe some unsuccessful early drafts, as well as the ultimate path of our design.

2.1 Goals

The first phase of the PADI wizard system included the following goals:

- Allow the interviewee to stop and start the wizard at any stage. That is, the wizard is not a "process funnel" that forces either completion or cancellation. Interviewees will be able to resume the interview if they choose to investigate some other path and then return to the same wizard page.
- Give status of completion at a glance.
- Provide easy navigation to any stage, step, or question within the interview.
- Integrate with the current interface in some reasonable way (i.e., avoid "modal" designs).
- Within the wizard system, store a "cardinality" for a given attribute within the wizard rather than in the model. For example, a "parent" *template* could have six Student Models, but the wizard should know that the end result of a wizard interview about a "child" *template* could have only one Student Model.
- Provide a wizard creation system so that steps within a wizard can be edited, rearranged, added, and deleted by the designer.

The original project team also discussed the ability for the wizard designer to impose cascading constraints, where an initial interview choice places constraints on other, subsequent choices in a dynamic, adaptive manner. For example, if the interviewee chooses to target a seventh-grade population, that selection might affect other choices, such as limiting the Materials and Presentation to be appropriate for that grade level. The interviewee would not even see inappropriate choices for Materials and Presentation, because those inappropriate choices would be hidden in an adaptive manner. This goal was discussed but put off for some future development phase.

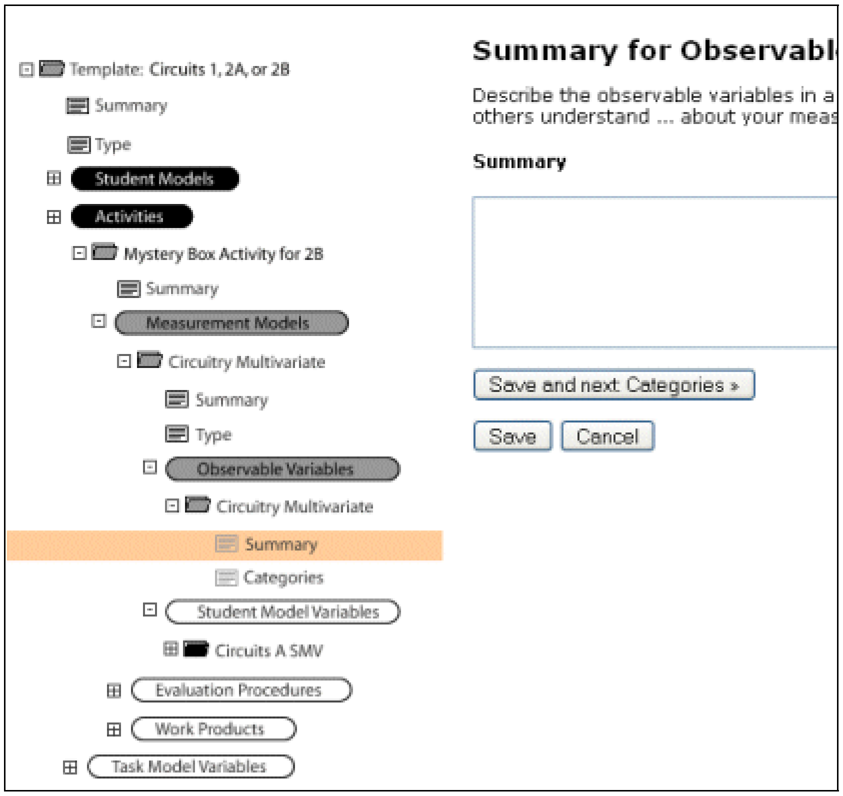
2.2 Early Attempts

Two early design attempts, an editing wizard and a simple sequential wizard, were postponed and repurposed, respectively.

2.2.1 An Editing Wizard Based on a Tree Display

At first, the design of PADI wizards focused on an editing scenario, where each and every piece of a *template* would be presented as an interview question. We imagined a tree display that would progressively expose pieces of the *template* in order to focus the interviewee's attention on a particular piece at one time while simultaneously providing context via the tree itself. Figure 2 shows an early mockup of the idea.

Figure 2. HTML Design of Editing Wizard



Describe the observable variables in a
others understand ... about your meas

In Figure 2, a tree view on the left side allows progressive disclosure of the contents of the *template* by expanding elements, starting at the root of the tree. The current focus in Figure 2 (highlighted in orange) is on the Summary attribute of an Observable Variable named "Circuitry Multivariate." Each selection of a leaf in the tree (on the left) causes the appropriate editor to appear on the right. The currently selected leaf, the Summary, displays an editor that happens to be a text box.

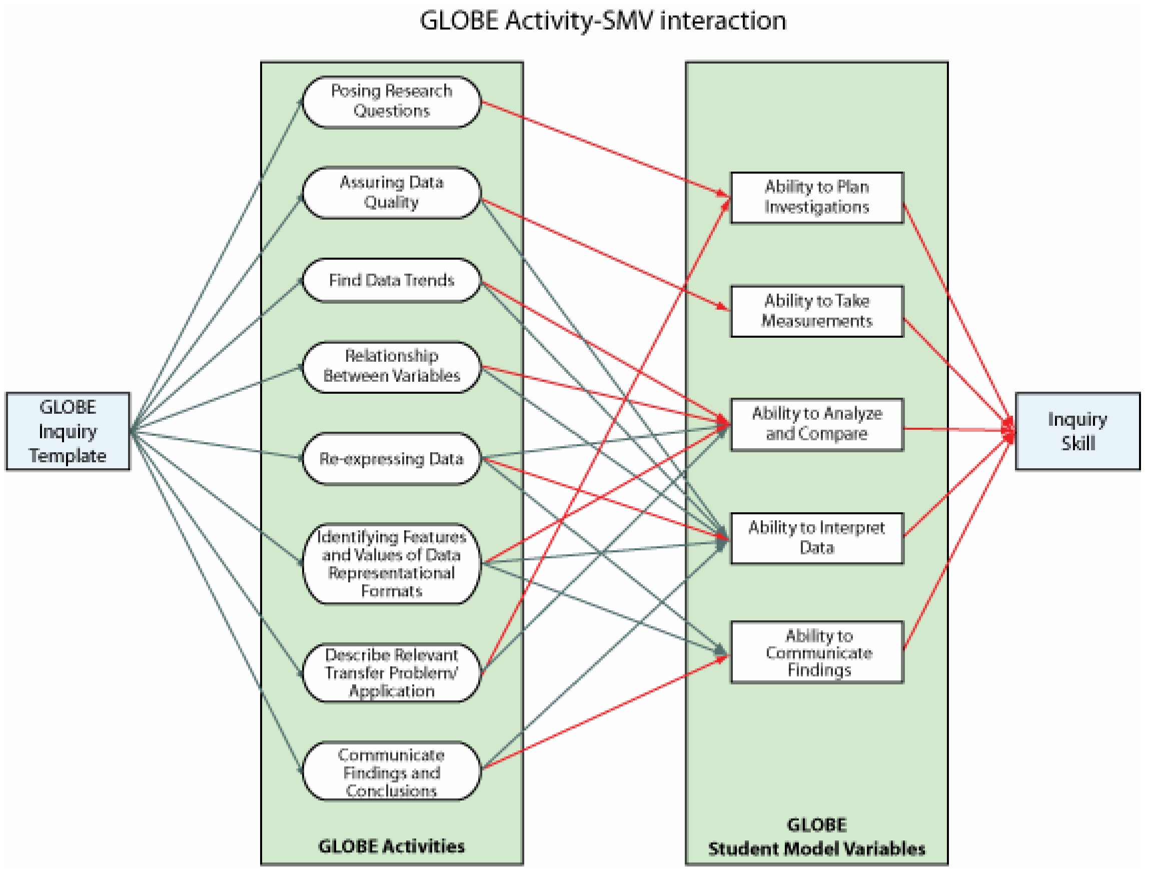
However, the Principal Investigators felt that the scope of this endeavor would be relatively large, so we sought a different way to scaffold the process of constructing a *template*. (This tree view was later completed in a development effort separate from the wizard effort.)

2.2.2 A Simple Sequential Wizard

In discussing how to assist relative novices in constructing a *template*, we first imagined a simple sequential process. The interviewee would start at the beginning of an interview, answer a question to fill in the first slot, and proceed to the next question. As we looked into how this would apply to our *templates*, however, we ran into problems with the constraints that initial questions could have on subsequent questions.

For example, in response to an initial question, if an interviewee's answer shows interest in constructing an assessment for the summative evaluation of an elementary school class, any subsequent questions regarding presentation materials should be geared to a summative assessment of elementary school students. That first answer should frame subsequent questions and potentially eliminate certain things like advanced activities that would be inappropriate for an elementary assessment. The cascading effect of the influence of initial questions and their answers could lead to a combinatorial explosion of changes. A PADI team member, Toshihiro Fumoto, drew diagrams to show how the initial questions of a GLOBE investigation (<http://www.globeassessment.sri.com/ assessments.html>) should cause other questions to be omitted, reframed, and otherwise dynamically manipulated within an interview.

Figure 3. GLOBE Interaction Between Activities and Student Model Variables


GLOBE Activity-SMV interaction
GLOBE Inquiry Template
Posing Research Questions
Assuring Data Quality
Find Data Trends
Relationship Between Variables
Re-expressing Data
Identifying Features and Values of Data Representational Formats
Describe Relevant Transfer Problem/ Application
Communicate Findings and Conclusions
GLOBE Activities
Ability to Plan Investigations
Ability to Take Measurements
Ability to Analyze and Compare
Ability to Interpret Data
Ability to Communicate Findings
GLOBE Student Model Variables
Inquiry Skill

In one of Fumoto's diagrams, shown in Figure 3, a column of activities in the green box on the left gives information about various Student Model Variables (SMVs) in the green box on the right. These relationships are depicted with blue and red arrows, with red indicating the strongest relationship. In a GLOBE assessment, this diagram shows that measuring a particular ability (SMV) depends on the choice of Activities. So when you first choose a goal of measuring certain abilities, choices of Activities are constrained. The first choice cascades into constraints on subsequent choices. A simple sequential wizard would fail if it did not constrain subsequent interview questions properly.

Thus, we felt that a simple, sequential wizard would underperform relative to our goals and repurposed this design for some later situation wherein we could guarantee that there were no cascading constraints. (Much later, we picked up this design as a "completion wizard" that could follow a selection wizard, as described in Section 5.3 below.)

2.3 A Selection Wizard

As we discussed the requirements of a complex assessment, it was clear that both the number and type of constraints were unbounded. It would take a great deal of scripting for a software wizard to determine and enforce the constraints that could cascade from early choices. And implementation that permitted a designer to describe the various constraints would require some kind of scripting system with Boolean logic and advanced features to handle all sorts of complexities. A sample rule from Fumoto's analysis above would be that whenever a particular SMV was chosen, certain Activities were implicated: one or more of the Activities would need to be performed in order to get information about the ability. A scripting system that accommodated this sample rule would have to take, as input, an initial choice of an SMV and allow the interviewee to make a mapping within the scripting system that resembles Fumoto's diagram above. This would require a flexible means of specifying a universe of possible choices (in our example, specifying Activities) and then specifying a mapping between a given criterion and individual items in the universe of possible choices. It would be a nontrivial system to implement.

At that point, Professor Mislevy, a Principal Investigator of the PADI project, deduced that we could preconfigure a set of *templates*, one for each possible permutation of the criteria. These *templates* already would be populated and constrained appropriately for each permutation of criteria. The work of the wizard would then simply be to select within the set of preconfigured *templates*. In that case, there would be no limit on the complexity of the configuration. Each permutation could have any number of cascading constraints, because we can construct a given *template* by hand, with complete knowledge of the criteria.

For example, consider three questions for three separate criteria (with possible answers in parentheses):

- What is the goal of the assessment? (Formative/Summative)
- Grade level of the students? (1–4/5–8/9–12)
- What is the content area? (Atmosphere/Hydrology/Soils/Earth Systems/Land Cover/Visualizations)

These criteria provide a matrix of 2×3×6 = 36 possible combinations (permutations), as indicated in Table 2.

Table 2. Permutations Matrix

Cell	Goal	Grade	Content
0)	Formative	Grade 1–4	Atmosphere
1)	Formative	Grade 1–4	Hydrology
2)	Formative	Grade 1–4	Soils
3)	Formative	Grade 1–4	Earth Systems
…	…	…	…
34)	Summative	Grade 9–12	Land Cover
35)	Summative	Grade 9–12	Visualizations

Mislevy suggested placing a preconfigured *template* at each of the cells above, and then the wizard criteria questions would select from among these 36 *templates*.

Consider the case where a designer undergoing the interview chooses Summative, Grade 9–12, Visualizations. Only one cell contains those criteria: cell 35. So the *template* that populates cell 35 will contain whatever constraints cascade from choosing a summative assessment of visualizations for high school students. This *template* will have no Activities for elementary school, none for Soils, and so on. It is specialized. The constraints can be as numerous or complex as necessary because we have complete control over the *template* that we preconfigure for the designer's selection. Given that there is no deterministic algorithm to specify in scripting to create the *template* from scratch, neither a scripting system nor training in such scripting is required.

In practice, when the designer completes an interview and thereby indicates a set of criteria, a given *template* cell in the matrix is indicated, and this *template* is cloned and offered for further customization to the designer. By duplicating a given *template* within the set of preconfigured *templates*, the wizard provides a good foundational *template* to the designer who entered the criteria.

A selection wizard implies the creation of a growing number of preconfigured *templates*, which can be a problem. The same *template* can be reused for several cells, potentially, but this technique does imply the need for some kind of hierarchical inheritance to assist with creating and maintaining a large group of related, preconfigured *templates*.

3.0 Implementation

We discuss implementation details from two perspectives: interviewee and developer. First we describe how interviewees can find the proper wizard, then how they run it and interact with it. The second perspective describes how a senior assessment designer can develop a wizard, including some details about the types of interview questions (steps) and the matrix of references (decision matrix) that determine what the interviewee will be rewarded with after finishing a selection wizard.

3.1 The Interviewee's Perspective

An interviewee must first find and then interact with a wizard.

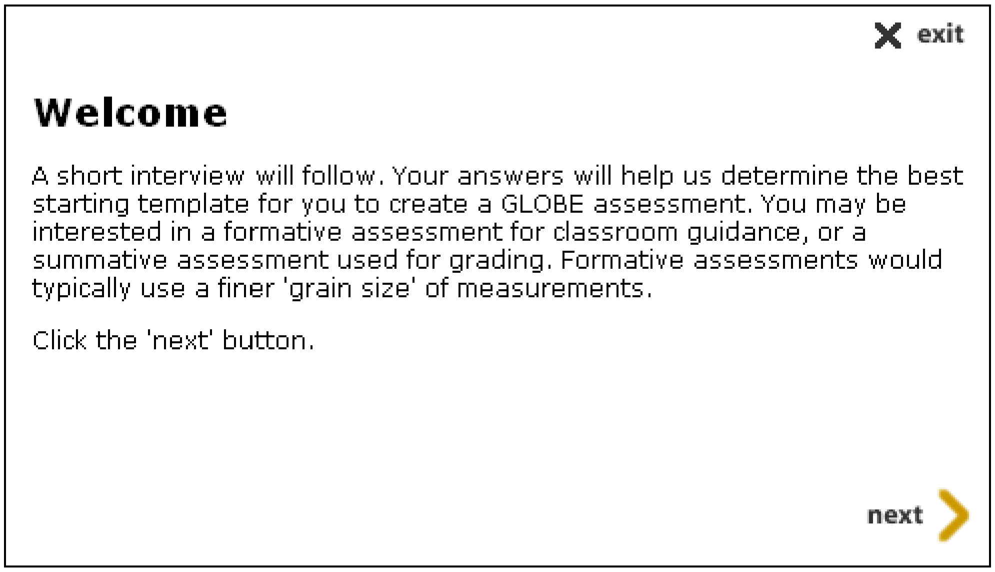
3.1.1 Finding the Wizard

Consider a scenario where a senior assessment designer has created some *templates* within the PADI design system and wishes to help more novice designers create assessments based on these *templates*. The senior designer could send a URL to the junior designers and ask them to create a new *template* based on the senior designer's *template*. Alternatively, a senior designer could create a selection wizard with a few criteria questions that help scaffold the experience of the junior designers. We envision a scenario where most of the "heavy lifting" has been done by a senior designer, including some of the difficult psychometric tasks associated with the Measurement Model. In this scenario, a senior designer would send a URL for the wizard, rather than one for the complex *template*. In addition, the design system provides a list of existing wizards on the front page to encourage exploration by novices, even if they have not directly received a link in the mail.

3.1.2 Running the Wizard

A wizard interview generally begins with a welcome screen, as shown in Figure 4.

Figure 4. The First Step of a Wizard



Welcome

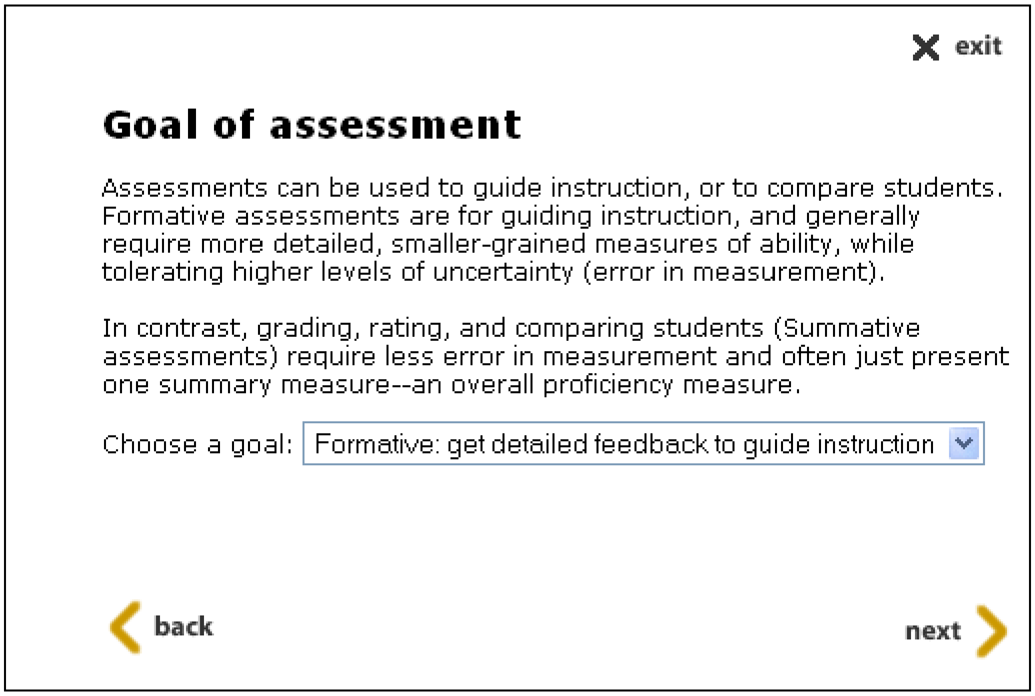
A short interview will follow. Your answers will help us determine the best starting template for you to create a GLOBE assessment. You may be interested in a formative assessment for classroom guidance, or a summative assessment used for grading. Formative assessments would typically use a finer 'grain size' of measurements.

Click the 'next' button.

This welcome step includes an explicit reference to the "next" button, just in case the interviewee is unfamiliar with Web pages. In the scenario above where the interviewee finds the wizard from a URL in an e-mail, this welcome page would be the first thing seen when clicking on the URL.

Next, any number of steps may follow. In our particular example wizard, the next step (see Figure 5) asks about the goal of the assessment, attempting to explain the choices and offering a menu selection.

Figure 5. A Step of an Interview



Goal of assessment

Assessments can be used to guide instruction, or to compare students. Formative assessments are for guiding instruction, and generally require more detailed, smaller-grained measures of ability, while tolerating higher levels of uncertainty (error in measurement).

In contrast, grading, rating, and comparing students (Summative assessments) require less error in measurement and often just present one summary measure--an overall proficiency measure.

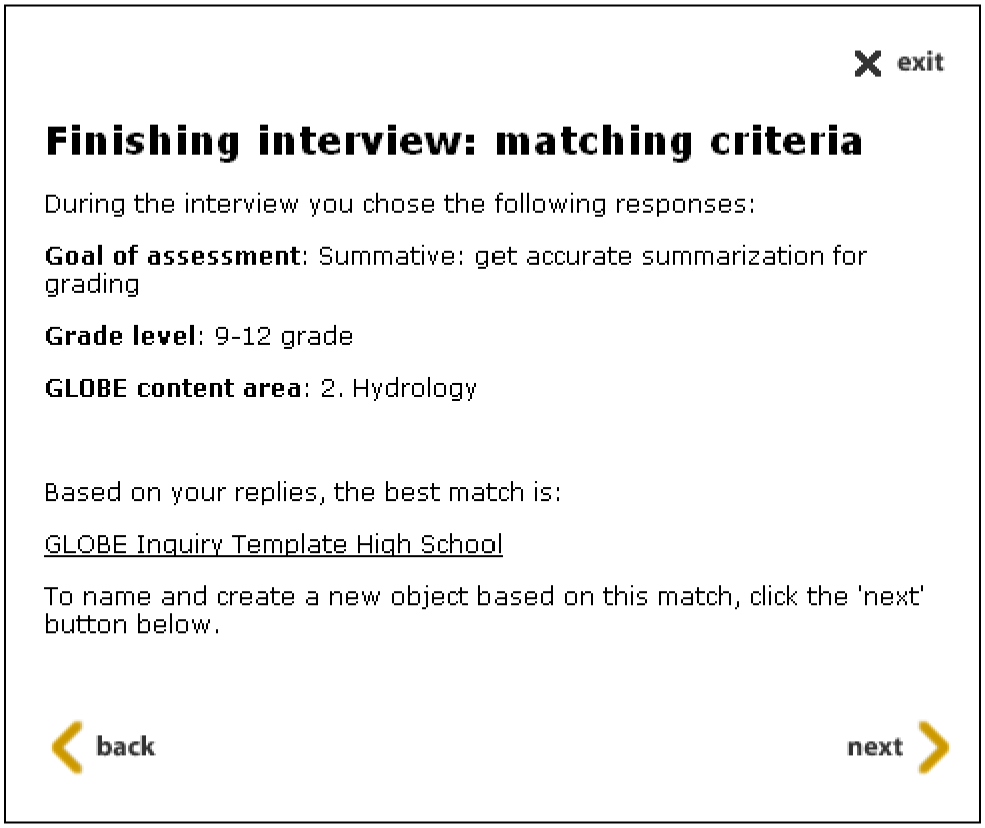
Choose a goal: Formative: get detailed feedback to guide instruction

Here the "back" button has appeared because there was a previous question. The interviewee has access to previous questions or can exit at any time. Using the browser's back button will cause the same behavior as the back button within the interview panel.

Entries Stored in Session

The information from the interviewee is held in session on the server and not saved to disk until the interviewee is finished, allowing the interviewee to exit and return to the interview while keeping the previous entries (during a session on the server currently set to expire after 30 minutes of inactivity). Only at the end of the interview is the interviewee prompted to clone a matching *template*, should the criteria have been matched, as shown in Figure 6.

Figure 6. Finishing Interview



Finishing interview: matching criteria

During the interview you chose the following responses:

Goal of assessment: Summative: get accurate summarization for grading

Grade level: 9-12 grade

GLOBE content area: 2. Hydrology

Based on your replies, the best match is:

GLOBE Inquiry Template High School

To name and create a new object based on this match, click the 'next' button below.

Should the interviewee's criteria find a match, as above, the next step offers to make a new copy of the matching *template*. The interviewee can also click on the title of the matching *template* to inspect it before (or instead of) cloning it.

In Section 3.2, we will change perspectives from the junior designer, or interviewee, to the person who develops the wizard.

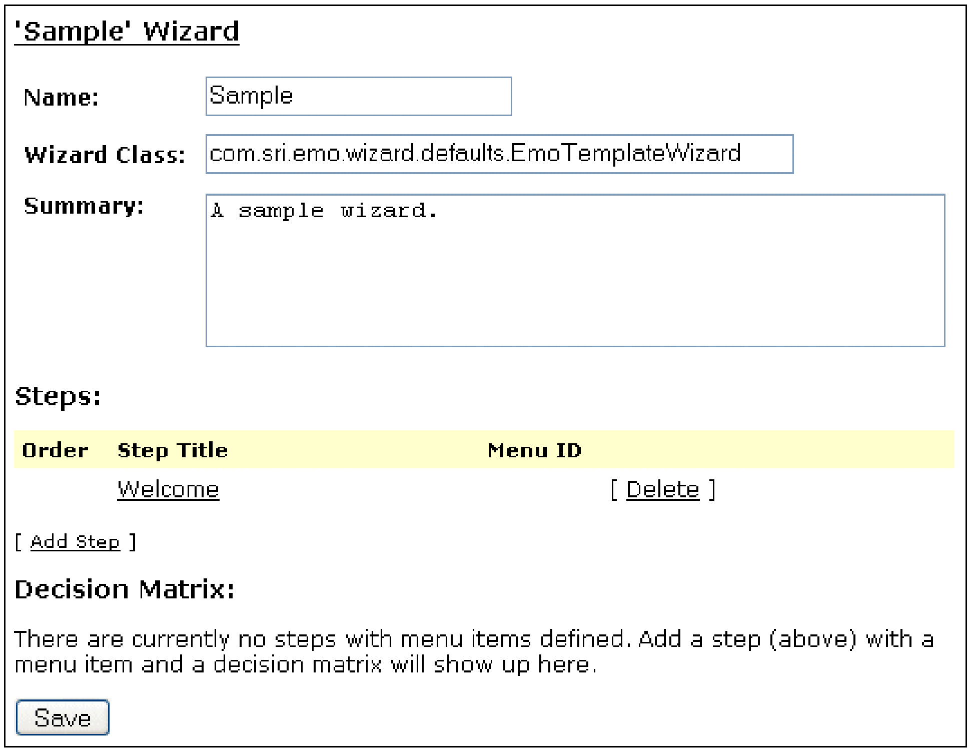
3.2 The Developer's Perspective

Our use-case revolves around a senior assessment designer with deep understanding of psychometrics who wishes to scaffold the process for more junior designers. Wizard creation is currently limited to the group of people who have permissions to edit metadata in the PADI design system. The initial page for creating a wizard is not visible without sufficient permissions.

3.2.1 A Wizard for Creating Wizards

Constructing a wizard is itself an interview process. An initial page asks for the name and description of the new wizard, along with a Java class that will serve as its runtime manager (a default class is provided). Next, an initial "Welcome" step is automatically added, as depicted in Figure 7.

Figure 7. Wizard Editing



'Sample' Wizard

Name:	Sample
Wizard Class:	com.sri.emo.wizard.defaults.EmoTemplateWizard
Summary:	A sample wizard.

Steps:

Order	Step Title	Menu ID	
	Welcome		[Delete]

[Add Step]

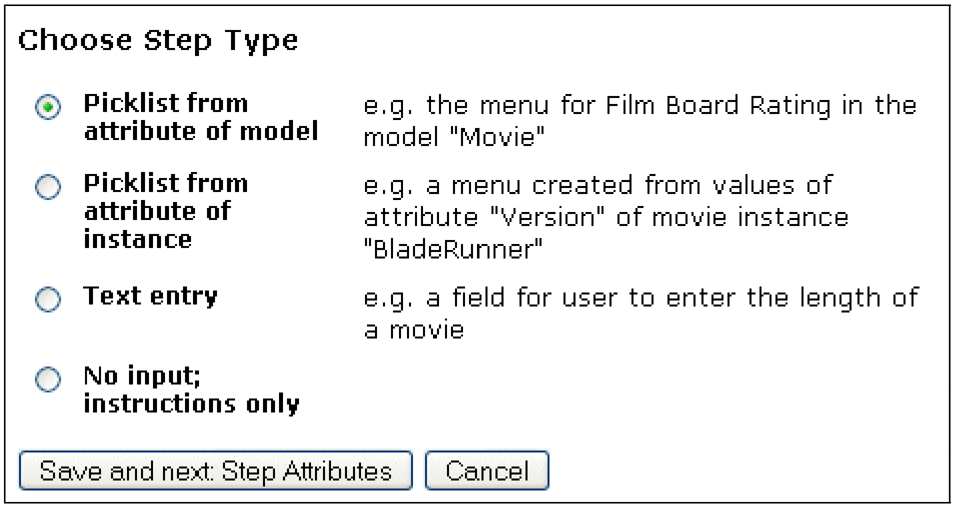
Decision Matrix:

There are currently no steps with menu items defined. Add a step (above) with a menu item and a decision matrix will show up here.

Here the word "step" means a single interaction (typically a single page) in the final wizard. In other words, the interviewee will interact with one step at a time, even though the developer may need to provide several pages of related information to specify a single step in the final product. There is no limit on the number of steps that can be added, although fewer are better, as described in Section 4.0.

There are several kinds of steps, as described below. The first question about creating a new step is to ask what kind of step is required, as shown in Figure 8.

Figure 8. Choosing Step Types



Choose Step Type

●	Picklist from attribute of model	e.g. the menu for Film Board Rating in the model "Movie"
○	Picklist from attribute of instance	e.g. a menu created from values of attribute "Version" of movie instance "BladeRunner"
○	Text entry	e.g. a field for user to enter the length of a movie
○	No input; instructions only	

Each type of step calls up a slightly different interview process.

3.2.2 Step Types

1. Picklist from attribute of model

A step that is constructed from an attribute of a model makes use of a menu already provided by the model. Here, "model" indicates an object definition in PADI. For example, a *template* is a data model containing various attributes. (When we name and fill in a particular *template*, we are filling in an instance of the *template* model.)

For example, consider the Measurement Model, which includes an attribute for "Measurement Model Type." The choices in this menu include:

- Dichotomous
- Partial credit
- Rating scale

Thus, this model already has a menu within the attribute for "Measurement Model Type." The "Picklist from attribute of model" type of wizard step simply supplies the interviewee with an interview question that presents a menu from the model, like the one shown above. After a designer chooses this kind of step for a wizard, the next question is to determine which model is required, and then which attribute (which menu, if multiple menus exist for the model) should be presented. Only attributes that are menus, like the sample "Measurement Model Type" above, are permitted to be selected for this kind of step.

2. Picklist from attribute of instance

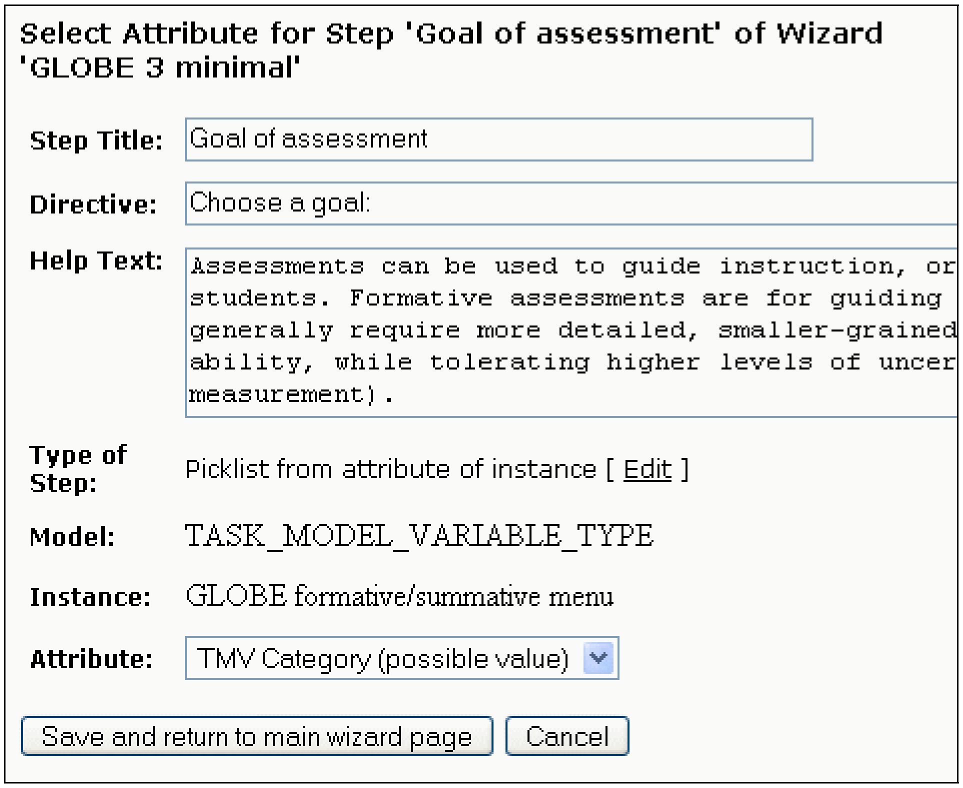
Similar to step type 1, a step type that is a picklist from an attribute of an instance will present a menu of choices when the wizard is run. In contrast to step type 1, the menu here is not presented as a menu in the model. Instead, a particular instance of the model, like a particular *template*, is identified, and within that particular instance, a particular set of attribute values is used to create a menu.

For example, a *template* named “EDMS 738 Assignments” has a Task Model Variable (TMV) called “topic area” that includes the following attribute values in its attribute called “TMV Category”:

1. Psychological underpinning
2. Statistical model
3. Task rationale

These attribute values can be made into a menu easily, as long as the wizard step knows both the instance ID and the attribute ID from which to construct the menu items. So after a designer chooses this kind of step for the wizard under construction, the next question is to determine the instance that is required and then the attribute required within that instance, as shown in Figure 9.

Figure 9. Wizard Step of Type “Picklist from Attribute of Instance”



Select Attribute for Step 'Goal of assessment' of Wizard 'GLOBE 3 minimal'

Step Title:	Goal of assessment
Directive:	Choose a goal:
Help Text:	Assessments can be used to guide instruction, or students. Formative assessments are for guiding generally require more detailed, smaller-grained ability, while tolerating higher levels of uncer measurement).
Type of Step:	Picklist from attribute of instance [Edit]
Model:	TASK_MODEL_VARIABLE_TYPE
Instance:	GLOBE formative/summative menu
Attribute:	TMV Category (possible value)

In Figure 9, a wizard step of this type already has been specified, along with the instance of the model. In this case, the instance is titled “GLOBE formative/summative menu.” In this screen, the wizard editing system has examined the instance and determined what attributes it has, and the system displays them in a menu labeled “Attribute.” The designer of the wizard simply chooses which attribute to use.

This type of step, creating a picklist from an instance, has turned out to be the most popular kind of step because of its flexibility. It is convenient to have complete control over what the menu will say for a given step in a wizard, and it is easy to construct a Task Model Variable or other simple instance in order to store the attribute values necessary for a custom menu. Furthermore, a selection wizard's design matrix allows arbitrary menu labels to be converted into contextualized decision criteria, as described in Section 3.2.3.

3. Text entry

A text entry step simply directs the interviewee to supply some text, given a directive and supporting descriptions. Text from any text entry step must be handled by custom code in a custom Java class specified for the wizard. Since a selection wizard typically is operated by making choices in a menu or otherwise, text entry steps typically will not exist in a selection wizard. An example use-case for a text entry step is one in which an interviewee is asked to enter a prompt for an assessment question, and the wizard (via the custom Java class) parses the text to provide some feedback about how well the interviewee's entry matches a prototype.

4. No input; instructions only

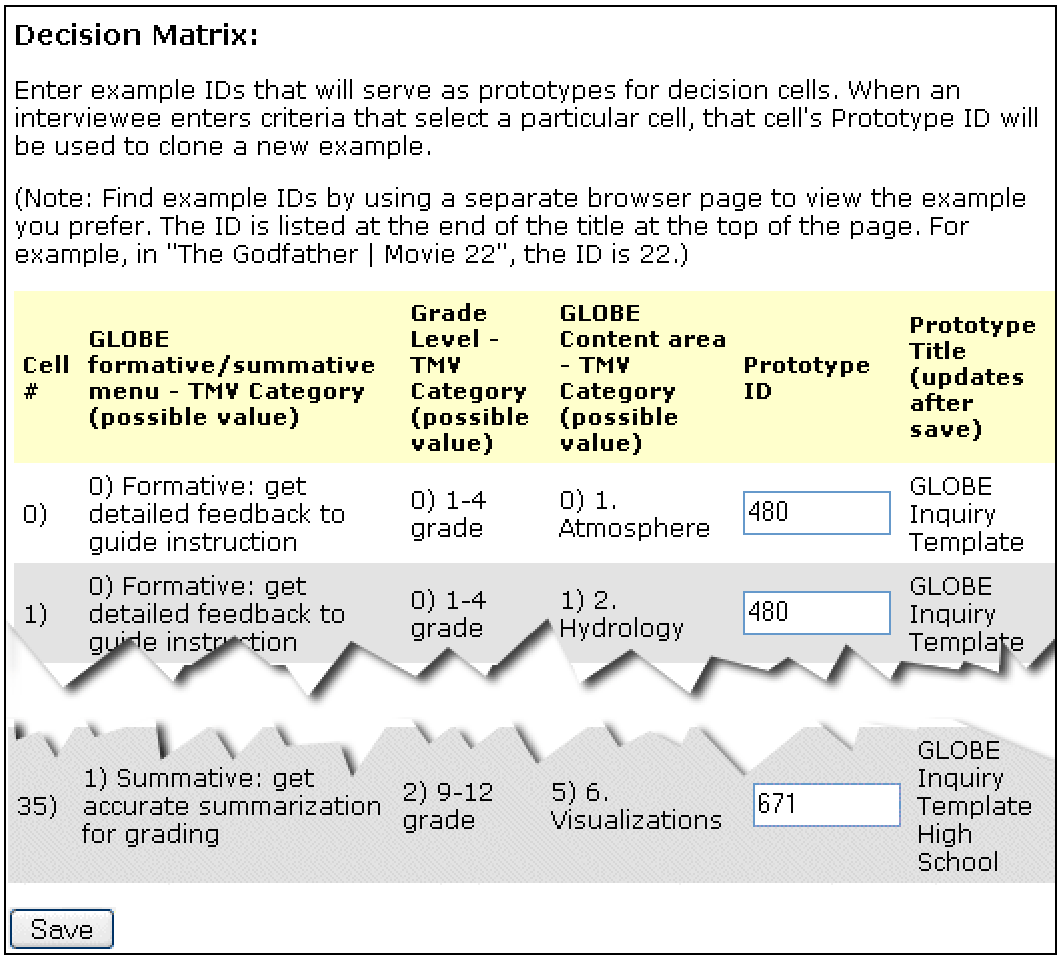
By default, each wizard starts with an instruction step: the welcome step. The welcome text is placed on the screen that the interviewee first sees. An instruction-only step has no input but only shows instructions and/or information. There is no special information necessary to add an instruction-only step to a wizard—just the text to be displayed.

3.2.3 Decision Matrix

At the bottom of Figure 7, the decision matrix is empty and therefore not shown. The decision matrix represents the reward at the end of each possible path that an interviewee can take. As the wizard prompts the interviewee through the prescribed series of steps, each menu-driven step adds a criterion (instruction steps are ignored), culminating in a set of criteria for selecting a *template*.

As steps are added by the developer, the system adds permutations based on those steps, and the decision matrix of the selection wizard gains rows and columns. The decision about which *template* to offer is determined by the entries made by the developer in the decision matrix. The developer places an ID for a *template* in each cell, as indicated in Figure 10.

Figure 10. Sample of a Decision Matrix



Decision Matrix:

Enter example IDs that will serve as prototypes for decision cells. When an interviewee enters criteria that select a particular cell, that cell's Prototype ID will be used to clone a new example.

(Note: Find example IDs by using a separate browser page to view the example you prefer. The ID is listed at the end of the title at the top of the page. For example, in "The Godfather | Movie 22", the ID is 22.)

Cell #	GLOBE formative/summative menu - TMV Category (possible value)	Grade Level - TMV Category (possible value)	GLOBE Content area - TMV Category (possible value)	Prototype ID	Prototype Title (updates after save)
0)	0) Formative: get detailed feedback to guide instruction	0) 1-4 grade	0) 1. Atmosphere	480	GLOBE Inquiry Template
1)	0) Formative: get detailed feedback to guide instruction	0) 1-4 grade	1) 2. Hydrology	480	GLOBE Inquiry Template
35)	1) Summative: get accurate summarization for grading	2) 9-12 grade	5) 6. Visualizations	671	GLOBE Inquiry Template High School

The decision matrix uses column labels appropriate for whatever attribute has been specified for a given step. In Figure 10, the first column label indicates the cell number (using zero-based counting). The next three column labels indicate the menus chosen for the first three steps: formative/summative, grade level, and content area, respectively. The next column is labeled Prototype ID because the instance (example) IDs entered into the wizard will be used to identify prototype *templates* that will be cloned. When an interviewee enters criteria that match a given cell, the Prototype ID of that cell is used to clone a new *template*. The last column of the decision matrix is the title of the *template* corresponding to the ID number entered by the designer of the wizard. This title will be shown only after entering and saving an ID (so that the system can retrieve the title). Likewise, if an ID is changed, the title will change after saving that new ID.

There is no requirement for Prototype IDs to be unique within a cell of the matrix, or even to be present. Several cells may use the same prototype or no prototype. In a case where a set of criteria leads to a cell with no prototype, the interviewee is told that there is no match for a particular set of criteria.

4.0 Discussion

4.1 Lessons from Initial Use of Prototype

As our research group made use of the initial wizard prototype, we found gaps and missteps that we addressed, as well as other requests that we could not accommodate. Feature requests were recorded in a list for discussion and prioritization. There has been no extensive use of the wizards yet, and none outside the research group. Nonetheless, our experts pointed out many issues as they used these tools.

4.1.1 Requirement for a Step with Ad Hoc Menu Leads to Abstract Way to Define Steps

The initial prototype included three types of wizard steps, but immediately there was a need for a fourth type of step that featured an ad hoc menu (displayed as step type 2 above). The engineers anticipated yet more types of steps in the future (for example, see the need for multiple-selection steps below), so they implemented an abstract means to define steps. In other words, it is now relatively easy to add a new type of step to the current list of four shown in Figure 8.

4.1.2 Request for Extensive Text and Graphic Content Within Wizard Step

Designers interacting with the wizard system sought to educate and scaffold interviewees within the context of a single wizard step. For example, if a wizard asks what the goal of an assessment is and offers the choices “Summative” and “Formative,” the wizard designer may wish to define those terms in great detail. The wizard designer may wish to include graphs and charts and pages of instruction. An opposing tension is that a wizard is intended to focus attention on one small item at a time and not distract the interviewee from that focus. To balance the tension, the wizard system supports putting URLs in the text of a wizard step. Such URLs can lead to any number of educational and lengthy explanations and have the benefit of being easily passed over by interviewees who do not need scaffolding. An alternative is some kind of pop-up help window, but this is not ideal for a Web browser situation where we typically seek to keep activities in a single window. See Section 4.2 for more deliberation on the tension between simplicity and detail in a wizard environment.

4.1.3 Summary Page

Wizard designers suggested that at the end of a wizard, a summary page reiterate all the choices made previously. This has since been incorporated.

4.1.4 Multiple Selection Instead of Single-Choice Menu

Wizard designers asked for a step in which more than one item could be chosen within a given step. For example, consider an assessment with several potential Activities, and the wizard requires a choice of one or more—multiple selection is possible. We could use multiple checkboxes to present multiple possible Activities. Again, the tension between simplicity and detail is evident because any multiple selection, in theory, can be broken into individual choices that are taken in sequence. In the example above, the multiple Activities could be presented in a single menu, and after making a single choice, the interviewee could choose to click a button to select yet another Activity or click a different button to finish and go on to the next part of the wizard. In this way, a multiple-selection situation would be translated into individual screens.

However, we agreed that there are situations when multiple simultaneous choices would be a superior interface, particularly when supported by checkboxes in a manner familiar to most interviewees. Such a feature is on the list of future enhancements for the wizard system.

4.1.5 Dynamic, Adaptive Steps That Change According to Previous Choices

Wizard designers asked for an adaptive wizard where the results of early steps affected the display and content of subsequent steps. See Section 4.3 for more deliberation on this topic.

4.1.6 Selection Versus Auto-Generating Wizard

Creating a selection wizard becomes a bigger challenge as the number of steps grows. More steps—more criteria—mean the requirement of more prepopulated matching *templates*. Wizard designers asked if any of these matching *templates* could be auto-generated.

It would be possible to auto-generate a *template* that matched a given set of criteria if that set of criteria did not impose cascading constraints. In the simplest case, the criteria are simply entered into a candidate *template*. Knowing whether a particular criterion depended on some other, previous criterion would require some kind of marking by the designer, a marking to indicate how criteria cascaded their constraints. A feature for auto-generating *templates* as prototypes for a given set of criteria is under consideration.

4.2 Simplifying

Wizards are supposed to simplify interaction for interviewees. TurboTax is supposed to make tax forms easier. We simplify especially for novices. Yet there can be benefits to giving the interviewee lots of control over the details, especially for more expert interviewees. So how much do we simplify—how many details do we decide for interviewees?

Wickham et al. (2002) stress the need to keep things simple, but they also discuss the tension between supporting experts versus novices. The authors suggest keeping a wizard designed for novices as simple as possible, while using the following techniques when the target audience includes more experts:

- Offer more detail that a novice may ignore by offering many entry fields that already have default values.
- Provide additional, optional pages at the end of the wizard for expert functions.
- Place expert functions in separate dialogs, moved behind "advanced" buttons.
- Design two paths through the wizard; one for experts, one for novices.

Another way to study the question of simplicity versus detail is to construct parallel wizards, one simple and the other more detailed, and to evaluate how interviewees interact with each. The results might tell us that one form is far preferred by all types of interviewees, that the forms are relatively equivalent, or that some audiences benefit from simplicity while others benefit from controlling the details.

We have constructed parallel wizards, with some on the simplified, assumption-making end of the scale and others on the more complex, ask-the-interviewee end of the scale. Again, we have no pool of interviewees to test, nor any mandate to attempt such testing. Instead, we must rely on expert opinion for now: the expertise of the research team in determining what seems to be the most beneficial design.

Considering the Wickham advice for two paths through the wizard, we anticipate that both our parallel (simple and complex) wizards could be used in concert if we can discern and separate the appropriate audiences for each wizard. We imagine creating an initial wizard that asks for a self-rating or otherwise gathers audience characteristics in order to determine which kind of wizard to launch next. One problem with discerning audience characteristics is that many novice interviewees may not know what they do not know; they may answer a metaquestion about their knowledge incorrectly. Another concern is redundancy. Having multiple wizards to achieve the same selection goal means additional software creation and maintenance.

Another potential redundancy concerns the intersection between the selection options in a wizard and the options available in the actual, resulting *template*. The flexibility that interviewees have *after* they finish the selection wizard could obviate the need to provide details during the wizard interview. Once interviewees are directed to a *template*, they can go back within the *template* to change whatever they like. Thus, it could be redundant to have interviewees answer many detail questions during the interview if they have control over details in the *template* after it is selected by the wizard.

We continue to deliberate on how to best resolve the tension between the requirement for simplicity and the requirement for sufficient detail.

4.3 Dynamic, Adaptive Steps Within a Wizard

During our discussions of wizard construction, it was clear that team members wanted more intelligence in the system. Such intelligence could allow the wizard to adapt when answers to early questions impact future questions. This is similar to the motivation for creating a selection wizard. In the selection wizard case, interviewees' criteria cause cascading constraints on the *template*. Similarly, answers to initial questions can impact future questions. For example, if the designer specifies a high school audience, it might be appropriate to hide an option that is only appropriate for elementary school audiences.

As with the discussion of the selection wizard, the implementation difficulty here is the open-ended requirement for flexibility. It would be relatively simple to specify whether or not to show an entire subsequent question based on the answer to a previous question, but the typical use-cases involve modifying items and branching down different paths based on previous answers. Tidwell and Fuccella (1997) created a scripting system for branching, but the use-cases described in our discussions go beyond branching into specific assessment logic. We would need a powerful scripting system to accomplish this branching, at which point the scripting becomes extremely complex. Cypher (1993) says that typical end users are not very interested in learning programming languages. We further discuss adaptive wizards in the next section.

5.0 Future Directions

There are many directions and further steps we could take in this research. Below are a few efforts that we prioritized as important.

5.1 User Testing

The motivation for creating wizards is to scaffold the experience of relatively novice interviewees. This scaffolding can be provided only through the efforts of experienced interviewees, so we quickly run out of test subjects within our PADI team who might qualify as inexperienced interviewees. In the future, recruiting a number of novices will help us understand where the wizards succeed and fail.

5.2 Revisiting the Priority for Dynamic, Adaptive Steps

When we started, we imagined an initial development phase to create basic wizards and a second phase to create wizards that supported dynamic, adaptive steps. These adaptive wizards would be relatively easy to do as a one-shot prototype, a special case, where all of the logic is explained to a programmer who then produces a one-of-a-kind wizard. To make such adaptation available in a general way, so that every assessment designer could specify all the logic without Java programming, we would need to add some kind of scripting ability—a kind of end-user programming. DiGiano, Kahn, Cypher, and Smith (2001) suggest that end-user programming is a large task and that we will want to have extensive educational supports within such a system.

Instead, we have provided a selection wizard that enables us to handle many of the complex interdependencies of *template* construction by preconfiguring the prototype *templates* for each cell of the decision matrix. In many cases, we can avoid an adaptive wizard by using the selection wizard for the cascading constraints that cause the complexity requiring adaptability. Thus, we have lowered the priority of creating an adaptive wizard.

5.3 Completion Wizard

To complement the selection wizard, we are now designing a prototype for a completion wizard, where various fields left incomplete by the selection wizard are scaffolded for the novice interviewee. In other words, after the selection wizard handles the complex parts of *template* construction that include cascading constraints, there may remain text fields and multiple-choice fields to be specified in the target *template*.

For example, consider a *template* where the choice of the student model will affect whether certain activities should be included or excluded. This kind of constraint calls for the use of a selection wizard. An interviewee interacts with the selection wizard, selecting a certain student model. That choice constrains the subsequent activities choices to, say, elementary school activities. But are all fields constrained? Say that the *template* has an additional, uncompleted field for specifying the first activity to be presented, as well as a text field for describing the materials to be used in the assessment. These fields have no cascading constraints based on their values. They are not suitable for a selection wizard that is only concerned with a few criteria questions of cascading impact. Furthermore, the selection wizard cannot handle an open-ended question like a text field for a description—that would require an infinite number of matching prototype cells for the infinite possible responses. On the other hand, a completion wizard could show a menu of the possible activities, with a directive to choose the first one to be presented. This same completion wizard could also present the text field for the description of the materials.

Thus, after getting the complex part of *template* construction done with the selection wizard, a completion wizard can follow to assist the interviewee in filling out the more mundane parts of the *template*.

References

Cypher, A. (1993). *Watch what I do: Programming by demonstration*. Cambridge, MA: MIT Press.

DiGiano, C., Kahn, K., Cypher, A., & Smith, D. C. (2001). Integrating learning supports into the design of visual programming systems. *Journal of Visual Languages and Computing*, *12*, 501–524.

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

Tidwell, D., & Fuccella, J. (1997). TaskGuides: Instant wizards on the Web. In *Proceedings of the 15th annual International Conference on Computer Documentation* (pp. 263–272). Salt Lake City, UT: ACM Press.

Wickham, D., Mayhew, D., Stoll, T., Toley, K., & Rouiller, S. (2002). *Designing effective wizards: A multidisciplinary approach*. Upper Saddle River, NJ: Prentice-Hall.

Sponsor
The National Science Foundation, Grant REC-0129331

Prime Grantee
SRI International. *Center for Technology in Learning*

Subgrantees
University of Maryland
University of California, Berkeley. *Berkeley Evaluation & Assessment Research (BEAR) Center* and *The Full Option Science System (FOSS)*
University of Michigan. *BioKIDs*