A Wizard for PADI Assessment Design

By Lawrence Hamel and Patricia Schank

April, 2005

Table of contents

Table of figures

Abstract

The Principal Assessment Design 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 (occasionally incomprehensible) constraints. To mediate the complexity, popular tax software provides an interview format, where filling out the form is reduced to answering of 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 designer who uses the system.

This paper includes a brief 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.

Background

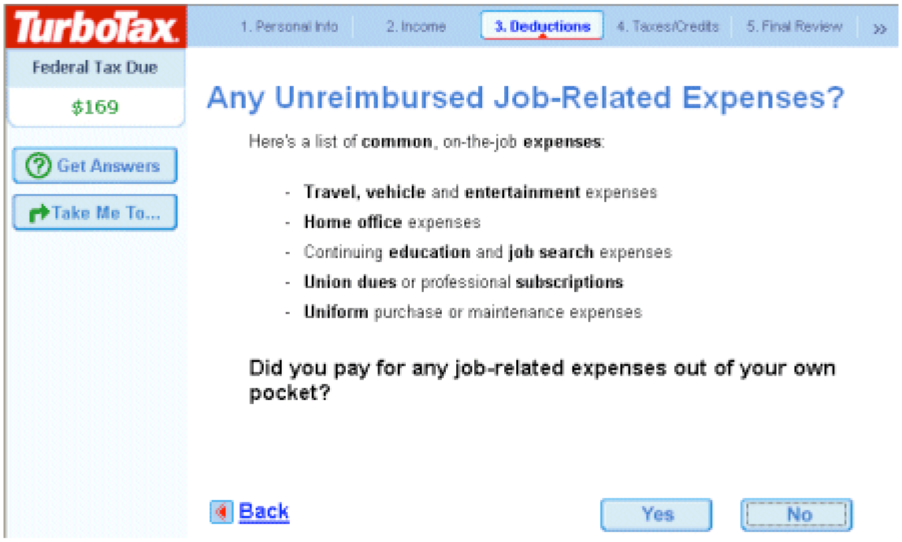
PADI

The Principal Assessment Design for Inquiry (PADI) project [http://padi.sri.com] is lead 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, delivering, and scoring complex assessment tasks data model for assessments. The conceptual framework defines a system of interrelated objects which 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 their own sets of attributes. It can be daunting to complete a Template from scratch.

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 (occasionally incomprehensible) 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.

Figure 1. TurboTax sample interview question



Any Unreimbursed Job-Related Expenses?

Here's a list of **common**, on-the-job **expenses**:

- **Travel, vehicle** and **entertainment** expenses
- **Home office** expenses
- Continuing **education** and **job search** expenses
- **Union dues** or professional **subscriptions**
- **Uniform** purchase or maintenance expenses

Did you pay for any job-related expenses out of your own pocket?

In Figure 1, there is 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 the menu at top (1. Personal Info, 2. Income, etc.). 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 (upper left) and a Help menu (not shown).

Likewise, the PADI Design System has a means to create and conduct interviews, also known as wizards, which prompt for decisions about assessment design.

Wizards in the literature

Wickham et al. (2002) describe a rationale and method for creating wizards, including the following general goals of wizard design (page 89):

- Hide the complexity of the underlying task.
- Ask the users simple questions.

- Include as many defaults as possible so users don't 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 deciding whether to integrate expert and novice functions within an integrated wizard, perhaps providing optional controls, optional pages, and dialogs that proceed from a button labeled "advanced."

Tidwell and Fuccella (1997) describe a system which uses SGML to provide a flexible wizard creation system called TaskGuides. It also has wizard of wizard functionality -- an interview process that can create an interview process. They offer a syntax which permits branching and other complex behavior.

The Design Process

Goals

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

- 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. Users shall 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 should have only one student model.

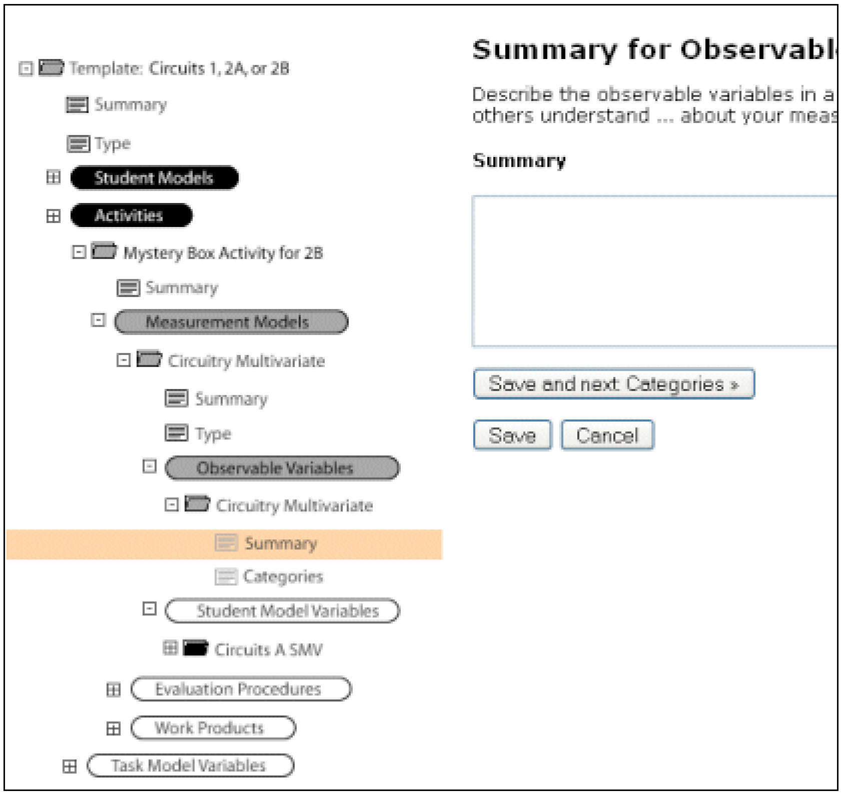
- Provide a wizard-creation system so that steps within a wizard can be edited, rearranged, added, and deleted by the designer.

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 were hidden in an adaptive manner. This goal was discussed but put off for some future development phase.

Postponing an editing wizard

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 view which would progressively expose pieces of the Template in order to focus the user's attention on a particular piece at one time. Below is an early mockup of the idea.

Figure 2. HTML Design of Editing Wizard



Summary for Observabl

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

Summary

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 which is 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.

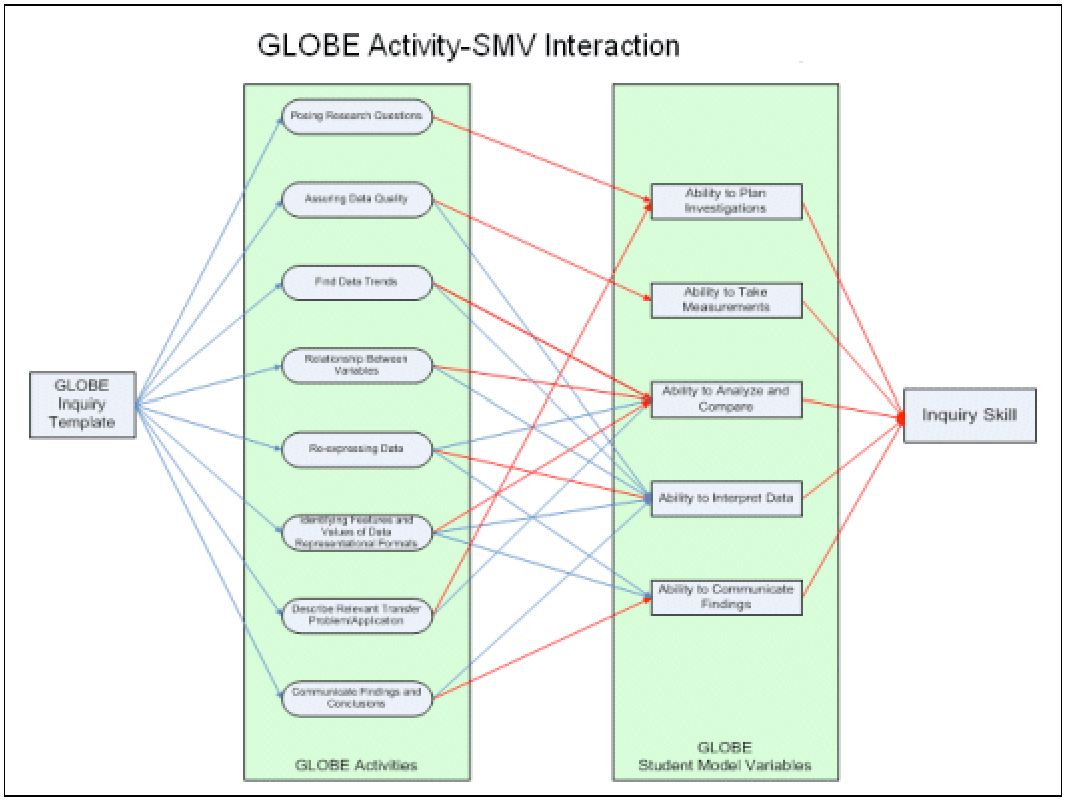
However, the principal investigators felt that the scope of this endeavor would be relatively large, so we sought to find a way to scaffold the process of constructing a Template in a different way.

Underachieving with a simple sequential wizard

In discussing how to assist relative novices in constructing a Template, we first imagined a simple sequential process. The assessment designer 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 assessment designer answers that they are interested 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/] 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. In a Globe assessment, this diagram shows that measuring a particular ability (SMV) depends on the choice of activities. So if 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.

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 very 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 user 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 criteria 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 pre-configure a set of Templates, one for each possible permutation of the criteria. These Templates would already be populated and constrained appropriately for each permutation of criteria. The work of the wizard would then be to simply 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 that content area? (Atmosphere/Hydrology/Soils/Earth Systems/Land Cover/Visualizations)

These criteria provide a matrix of 2x3x6 = 36 possible combinations (permutations):

Table 1, Permutations matrix.

Cell #	Goal	Grade	Content
0)	Formative	elem 1-4	Atmosphere
1)	Formative	elem 1-4	Hydrology
2)	Formative	elem 1-4	Soils

3)	Formative	elem 1-4	Earth Systems
…	…	…	…
34)	Summative	high 9-12	Land Cover
35)	Summative	high 9-12	Visualizations

Mislevy suggested placing a preconfigured Template at each of the cells above, and then the Wizard criteria questions would just select from among these 36 Templates.

Consider the case where a designer undergoing the interview chooses: Summative, high 9-12, Visualization. Only one cell contains those criteria: cell 35. So the template that populates cell 35 will contain whatever constraints that should cascade from choosing a summative assessment of visualizations for high school students. This Template will have no activities for elementary school, none for Soils, etc. It is specialized. The constraints can be as numerous as necessary, and as complex, because we have complete control over the Template that we pre-configure for the designer's selection. Since 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.

Implementation

Finding the wizard

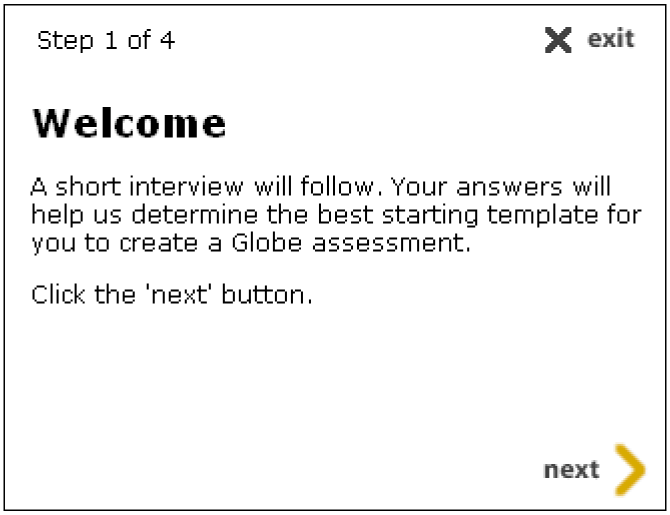
Consider a scenario where a senior assessment designer has created some Templates within the PADI Design System and wishes to help more novice designers create assessments based on her Templates. The senior designer could send a URL to the junior designers, and ask them to create a new Template based on to senior designer's Template. Alternatively, a senior designer could scaffold the experience more by creating 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.

Running the wizard

A wizard interview generally begins with a welcome screen as shown in the next figure.

Figure 4. The first step of a wizard



Step 1 of 4

Welcome

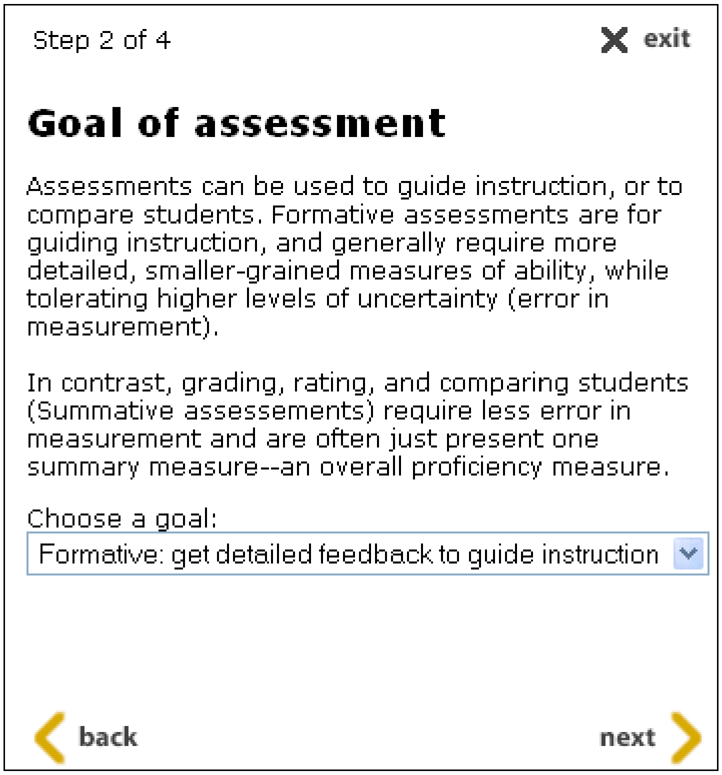
A short interview will follow. Your answers will help us determine the best starting template for you to create a Globe assessment.

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 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 assessements) require less error in measurement and are often just present one summary measure--an overall proficiency measure.

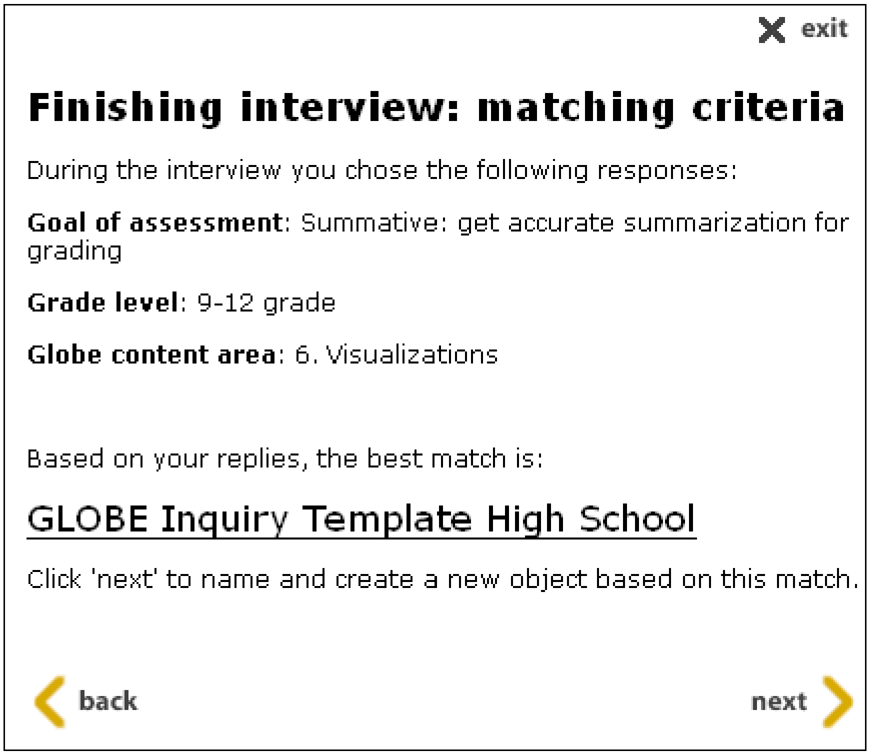
Choose a goal:
Formative: get detailed feedback to guide instruction

Here the "back" button has appeared since 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 (not saved to disk until finished)

The information from the interviewee is held in session on the server, allowing the interviewee to exit and return to the interview while keeping their previous entries (during a session on the server that is 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 their criteria have been matched, as shown in the next figure.

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: 6. Visualizations

Based on your replies, the best match is:

GLOBE Inquiry Template High School

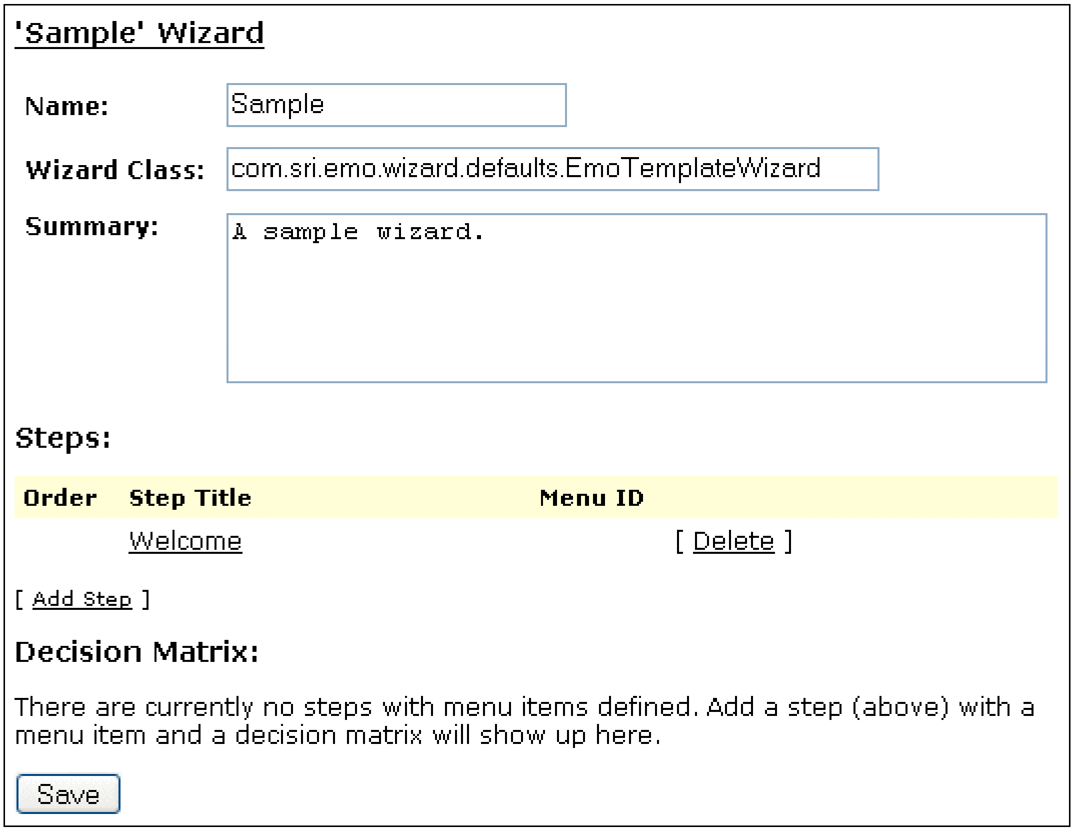
Click 'next' to name and create a new object based on this match.

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.

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 step "Welcome" is automatically added, as depicted in the next figure.

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]

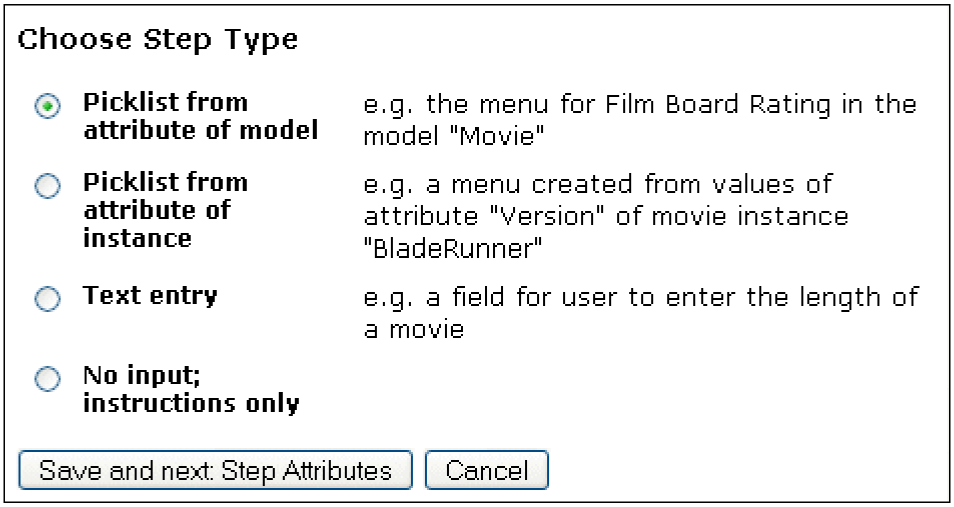
[Add Step]

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.

Additional steps can be added. There are several kinds of steps, and some of them require multiple interview questions to specify. So the first question about creating a new step is to ask what kind of step is required, as shown in the next figure.

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 has a slightly different interview process, described below.

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" is used to indicate what we typically call 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.)

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

- Dichotomous,
- Partial Credit,
- Rating scale

(etc.)

Thus, this model already has a menu, and this type of wizard step simply supplies the interviewee with an interview question that presents that same menu. So after a designer chooses this kind of step for the wizard under construction, the next question is to determine which model is required, and then which possible attribute (which menu, if multiple menus exist for the model) should be presented. Only attributes which are menus, like "Measurement Model Type" are permitted.

2. Picklist from attribute of instance

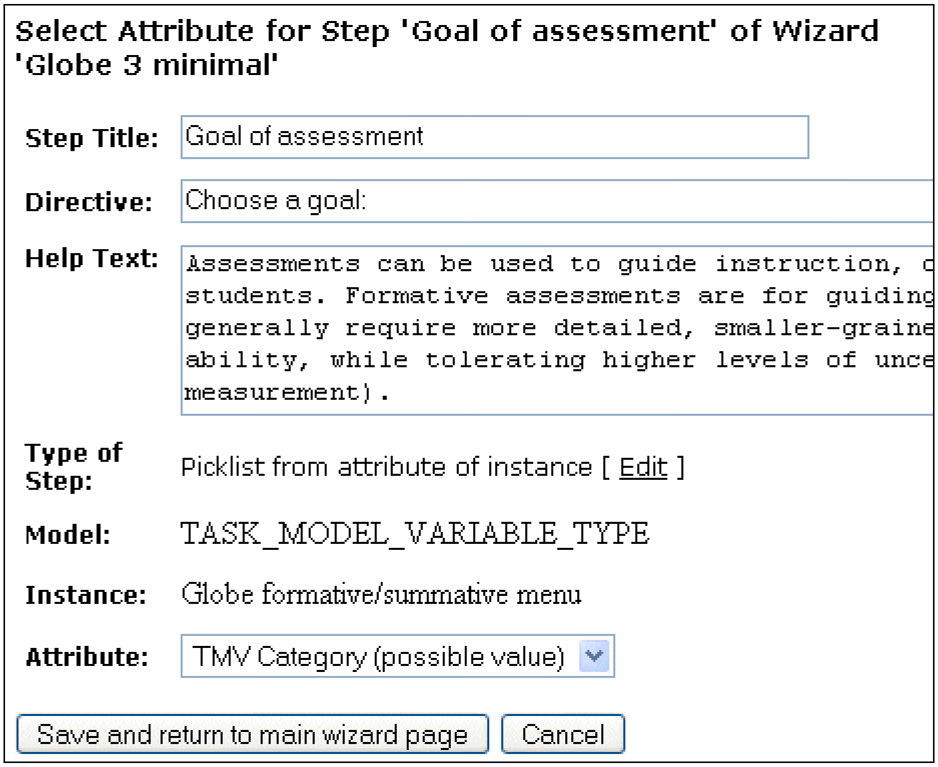
Similar to step type 1), a step type that is a picklist from an attribute of an instance will present a menu of choices when the wizard is run. In contrast to step type 1), the menu here is not presented as a menu in the model. Instead, a particular instance of the model, like a particular Template, 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 called "topic area" that includes the following attribute values in its attribute slot 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.

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, c students. Formative assessments are for guiding generally require more detailed, smaller-graine ability, while tolerating higher levels of unce 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 the figure above, a wizard step of this type has already 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 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 the next section.

3. Text entry

Text entry step is quite simple, simply directing the interviewee to supply some text, given a directive and supporting descriptions. There is no particular information necessary to add a text-entry step. However, the wizard must finally know what to do with the text supplied by the interviewee.

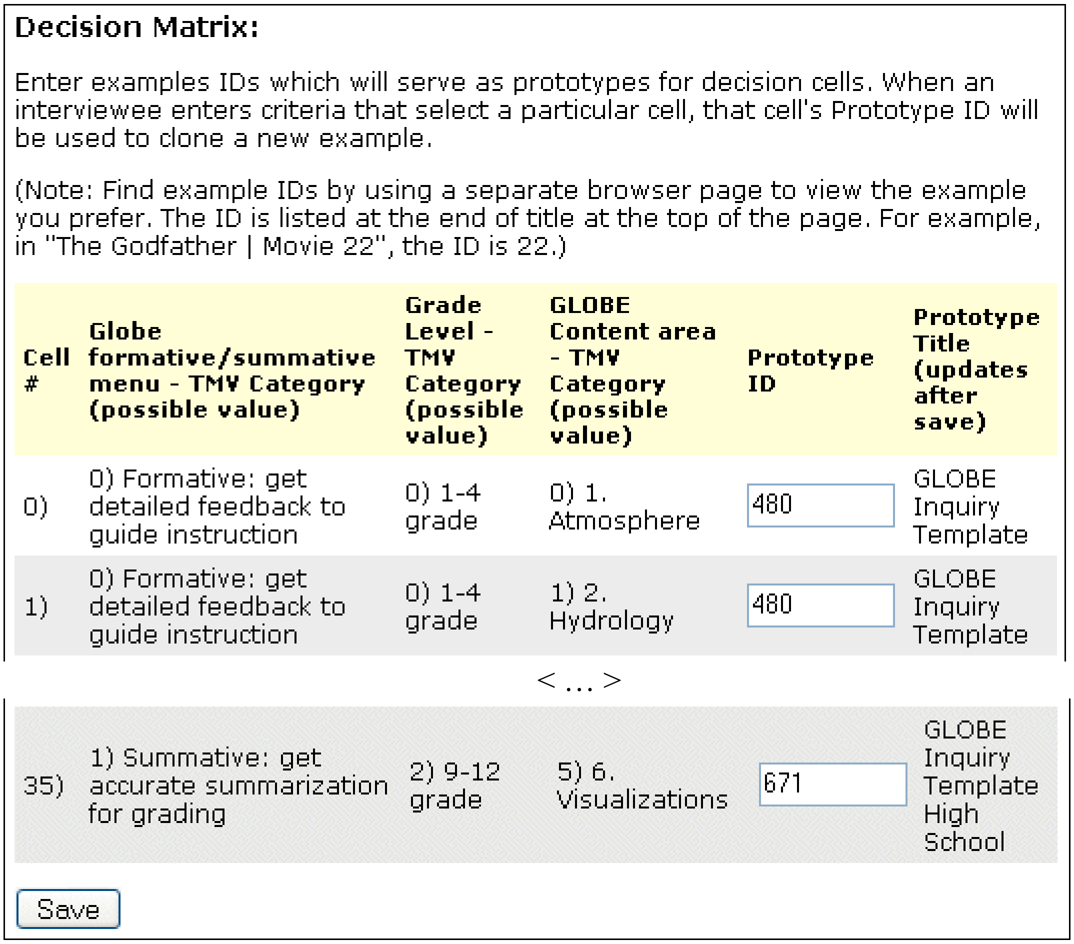
4. No input; instructions only

The welcome screen is an example of a step that has no input, but only shows instruction and/or information. There is no particular information necessary to add an instruction-only step to a wizard.

Decision matrix

At the bottom of Figure 7, the Decision Matrix is empty and therefore not shown. However, as steps are added, we add permutations to the decision matrix for the selection wizard. As mentioned above, the wizard prompts the interviewee through a series of steps which culminate in a set of criteria for selecting a Template. The decision about which Template to offer is determined by the entries in the Decision Matrix.

Figure 10. Decision Matrix



Decision Matrix:

Enter examples IDs which 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 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. 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) ID's 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 ID's 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 their particular set of criteria.

Discussion

Lessons from initial use of prototype

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 which 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.

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 URL’s in the text of a wizard step. Such URL’s 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 would be to have some kind of pop-up help window, but this is not ideal for a Web browser situation, where we typically seek to keep activities on a single window. See the Simplifying section below for more deliberation on the tension between simplicity and detail in a wizard environment.

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.

Multiple selection instead of single-choice menu

Wizard designers asked for a step in which more than one item should 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 possible. We could use multiple checkboxes to present multiple possible activities. Again, the tension between simplicity and detail is evident, since any multiple selection can, in theory, be broken into individual choices which 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 user 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.

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 the section below entitled “Dynamic, adaptive steps…” for more deliberation on this topic.

Selection vs. 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 pre-populated 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 which criteria cascaded constraints. A feature for auto-generating Templates as prototypes for a given set of criteria is under consideration.

Simplifying

Wizards are supposed to simplify, yet there can be a benefit to giving the user lots of control over the details.

So how much do we simplify--how many details do we decide for users?

Wickham et al. (2002) stress the need to keep things simple, but also have a chapter on the tension between supporting experts and novices. They suggest the following techniques to support experts.

- Offer detail via many detail entry fields that already have default values that a novice may ignore..
- 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 is to construct parallel wizards, one simple and the other more detailed, and evaluate the user experience. The research results might tell us that one form is far preferred, or 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-user end of the scale. However, we have not conducted an empirical study of user experience. We rely on the expertise of the research team in evaluating the most beneficial design.

Multiple 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 audiences is that many novice users to may not know what they don't know; they may answer a meta-question about their knowledge incorrectly. Another concern is redundancy. Having multiple wizards to achieve the same selection goal means additional software creation and maintenance.

Another redundancy concerns a particular feature of selection wizards. The flexibility that users have after they finish the selection wizard could obviate the need to provide details during the interview. After a selection wizard, users can go back within the template to change whatever they like. It could be redundant to have users 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.

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, user's 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 very specific assessment logic. We would need a powerful scripting system to accomplish this, at which point the scripting becomes very complex. Cypher (1993) says that typical end-users are not very interested in learning programming languages. See more discussion on adaptive wizards in the next section.

Future directions

User testing

The motivation for creating wizards is to scaffold the experience of relatively novice users. This scaffolding can only be provided through the efforts of experienced users, so there is a problem that we quickly run out of test subjects within our team who might qualify as an inexperienced user of the system.

In the future, recruiting a number of novice users to try out our various wizards will help us understand where they succeed and fail.

Revisiting the priority for dynamic, adaptive steps

When we started, we imagined a second development phase to create wizards that support dynamic, adaptive steps. This would be relatively easy to do for a special case, where all of the logic was explained to a programmer, who then produces a special-case, one-of-a-kind wizard. However, to make this procedure general so that the assessment designer might specify all the logic would be a kind of end-user programming, which is a large task. DiGiano et al. (2001) suggest that we will want to have extensive educational supports within such an end-user programming system, and otherwise put many resources into its construction.

Meanwhile, we have provided a selection wizard which enables us to handle many of the complex interdependencies by preconfiguring the prototype templates for each cell of the decision matrix. In many cases, we can avoid a complex wizard by using the selection wizard for the parts that are complex. Thus, we have re-prioritized the goal of creating an adaptive wizard.

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 user. The selection wizard handles the complex parts, leaving open-ended fields and more simple multiple-choice fields for the completion wizard. After all, the selection wizard has no chance of handling an open ended question that would require an infinite number of matching prototype cells. For example, consider a template where the choice of the student model will affect whether certain activities should be included or excluded. A selection wizard will help us narrow the field to, say, elementary school activities, and in a completion wizard can show a simple menu to choose among all the possible elementary school activities. The same completion wizard could also allow for open-ended questions, like asking for a specific description of materials and presentation that fit within the constraints of the rest of the Template.

References

Cypher, A. (1993). *Watch What I Do: Programming by Demonstration* MIT Press, Cambridge, MA.

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

Tidwell, D. and Fuccella, J. (1997) TaskGuides: instant wizards on the Web
ACM Special Interest Group for Design of Communications
Proceedings of the 15th annual international conference on Computer documentation
Salt Lake City, Utah. ACM Press. Pages: 263 – 272

Wickham, D., Mayhew, D., Stoll, T., Toley, K., Rouiller, S. (2002). *Designing Effective Wizards: A Multidisciplinary Approach* Prentice Hall