Technology Supports for Assessment Design

Robert J. Mislevy[1], Isaac I. Bejar[2], Randy E. Bennett[2],
Geneva D. Haertel[3], & Fielding I. Winters[1]

I. Introduction

Educational assessment concerns making inferences about what students know or can do, based on observing what they say, do, or make in a specified set of particular circumstances, i.e., tasks. Designing an effective assessment requires analyzing the domain, building an argument, and creating tasks that can be effectively administered and scored. This series of processes involves gathering, organizing, and transforming information in a variety of representational forms. At each stage, technology supports can organize work, enhance validity, and increase efficiency. This article discusses roles and provides examples of technology supports for assessment design. To frame the discussion, the following section reviews assessment arguments and layers in the design process.

II. Arguments, Layers, and Knowledge Representations

An active line of research in assessment is making explicit the principles that underlie educational assessment. This work provides structures and representations around which tools can be devised to improve and facilitate the efforts of assessment designers. Examples include Embretson's (1998) "cognitive assessment design system" and Luecht's (2002) "integrated test design, development and delivery." The present article uses the language of Mislevy, Steinberg, and Almond's (2003) "evidence-centered design" (ECD) approach. ECD has been implemented in assessment projects including Principled Assessment Designs in Inquiry (PADI), Cisco Systems' NETPASS simulation-based assessment, and ETS's work with the National Board of Professional Testing Standards, the Test of English as a Foreign Language (TOEFL), and the Information and Communication Technology (ICT) Literacy Assessment.

The central ideas in ECD are the assessment argument, layers of assessment, and the role of knowledge representations in designing and implementing assessments. Messick (1994, p. 16)

[1] University of Maryland, College Park
[2] ETS
[3] SRI International

orients assessment designers to the key aspects of an assessment argument by asking, “what complex of knowledge, skills, or other attributes should be assessed... Next, what behaviors or performances should reveal those constructs, and what tasks or situations should elicit those behaviors?”

Design typically begins with a purpose and some ideas about what is important in the domain, and then it moves to the specific materials and processes of an operational assessment. Adapting a “layers” metaphor from architecture and software engineering (Mislevy & Riconscente, 2006), ECD organizes the design process in terms of the following layers: domain analysis, domain modeling, conceptual assessment framework, assessment implementation, and assessment delivery. Table 1 summarizes these layers in terms of their roles, key entities (e.g., concepts and building-blocks), and knowledge representations that assist in achieving each layer’s purpose. The layering suggests a sequential design process, but cycles of iteration and refinement across layers are the norm.

The fundamental work in assessment design can be viewed as creating, transforming, and using information in the form of the knowledge representations within and between such layers.

Table 1. Layers of Evidence Centered Design for Educational Assessments

Layer	Role	Key Entities	Selected Knowledge Representations
Domain Analysis	Gather substantive information about the domain of interest that has direct implications for assessment: how knowledge is constructed, acquired, used, and communicated.	Domain concepts, terminology, tools, knowledge representations, analyses, situations of use, patterns of interaction.	Content standards, concept maps (e.g., Atlas of Science Literacy, AAAS, 2001). Representational forms and symbol systems of domain of interest, such as, algebraic notation, maps, computer interfaces.
Domain Modeling	Express assessment argument in narrative form based on information from Domain Analysis.	Knowledge, skills and abilities; characteristic and variable task features, potential work products, potential observations.	Assessment argument diagrams, PADI design patterns, content-by-process matrices
Conceptual Assessment Framework	Express assessment argument in structures and specifications for tasks and tests, evaluation procedures, measurement models.	Student, evidence, and task models; student model, observable, and task model variables; rubrics; measurement models; test assembly specifications.	Test specifications; algebraic and graphical representations of measurement models; PADI task template; item generation models; generic rubrics; algorithms for automated scoring.
Assessment Implementation	Implement assessment, including authoring presentation-ready tasks, scoring guides or automated evaluation procedures, and calibrated measurement models.	Task materials (including all materials, tools, affordances); pilot test data for honing evaluation procedures and fitting measurement models.	Coded algorithms for rendering tasks, interacting with examinees, evaluating work products; tasks as displayed; IMS/QTI representation of materials; ASCII files of item parameters.
Assessment Delivery	Coordinate interactions of students and tasks: task-and test-level scoring; reporting	Tasks as presented; work products as created; scores as evaluated.	Renderings of materials; numerical and graphical summaries for individual and groups; IMS/QTI results files

III. Supports for Domain Analysis

Domain-analysis is the first step in the process of designing and delivering an assessment. It entails gathering information about how people acquire, construct, represent, use, and communicate knowledge within the domain. It lays the foundation for later layers by defining the knowledge, skills, and abilities (KSAs) that assessment users want to make inferences about, student behaviors they can base their inferences on, and situations that will elicit those behaviors. This process is relevant to assessments of all kinds, whether formative or summative, large-scale or classroom.

Technology supports can be used in Domain Analysis to aid in defining and gathering the domain information, then organizing it in ways that will inform or facilitate task design. One well-studied example of technology tool that streamlines domain analysis is Shute, Torreano, and Willis's (2000) automated knowledge elicitation tool DNA (for Decompose, Network, Assess). DNA provides structured, user-friendly web forms to elicit domain experts' input on declarative, procedural, and conceptual-knowledge requirements of common tasks in the domain. Other examples include Williams's (2000) CAT-HCI software tool for eliciting experts' domain knowledge, and iGEN Cognitive Agent Software from CHI systems for analyzing and subsequently modeling cognitive tasks (Zachary, Taylor, & Hicinbothom, 2000). More specialized tools can be employed for specific purposes. Technology-based tools can be used to capture information about experts' knowledge that would be difficult to capture with more traditional methods of task analysis. For example, in the domain of satellite–image analysis, eye-trackers are used as part of cognitive task analyses (Kurland, Gertner, Bartee, Chisholm, & McQuade, 2005).

IV. Supports for Domain Modeling

Domain modeling structures the outcomes of domain analysis in a form that reflects the structure of an assessment argument, in order to ground the more technical student, evidence, and task models that are required in the subsequent Conceptual-Assessment Framework (CAF) layer. One tool for this purpose is *design patterns,* another concept that originated in architecture (Alexander, et al. 1977), and that was later adapted in computer science (Gamma, Helm, Johnson, & Vlissides, 1995). As defined by Alexander (1977), a design pattern provides a "description of a problem which occurs over and over again in our environment, and then

describes the core of the solution to that problem, in such a way that you can use this solution a million times over." Design patterns for assessment compile knowledge about ways to address assessment challenges that recur across domains or within particular domains (Mislevy, et al., 2003). Examples of design patterns from PADI include Model Revision and building a Scientific Explanation. Each makes explicit the knowledge and skills that the developer wants to measure, the kinds of observations that could provide evidence about the acquisition of the knowledge and skills, and features of task situations that allow the examinee to provide the requisite evidence. Hively, Patterson, & Page (1968) proposed a type of design pattern called *item forms* for assessing behavioral objectives. *Items models* (e.g., Bejar, et al., 2003) and *item structures* (Embretson, 1998) were developed to incorporate an accounting of task features from the perspective of their cognitive-processing demands.

Technology supports can facilitate the domain-modeling process by helping designers create, search, or tailor design patterns. In PADI, design patterns help designers bridge the educational goals (in the form of standards, learning objectives or knowledge, skills and abilities) with the technical design specifications that will be realized in an operational assessment.

Embodying a validity argument in an assessment requires more than authoring individual tasks. The assessment designer must judiciously use certain design patterns in ways and in combinations that best support the purpose of the assessment as a whole (see, for example, Davidson & Lynch, 2001, on test specifications). Many considerations, including resources, operational constraints, content representation, and statistical information, impact how many and which design patterns can be used to produce an *optimal* design. Optimization tools that have been developed for test assembly (van der Linden, 2005) can be used to sort through competing sets of design patterns that would lead to an assessment satisfying an array of constraints, including the adequacy of coverage of KSAs or educational standards.

V. Supports for the CAF

The Conceptual Assessment Framework (CAF) concerns technical specifications for the nuts and bolts of assessments, that is, the materials and processes that embody assessments. The central models for task design are the student model, evidence model, and task model (Figure 1). An assessment argument laid out in narrative form at the domain-modeling layer is now expressed in terms of specifications for tasks, measurement models, scoring methods, and

delivery requirements. Details about task features, measurement model parameters, stimulus material specifications, and the like are expressed in terms of representations and data structures that will guide their implementation and ensure their coordination. Technology supports can be applied with each of the CAF models to facilitate task designers' work. Moreover, knowing the supports that are available at the level of the CAF helps shape the designer's work in domain analysis and domain modeling. The kinds of supports that are available to implement and carry out assessments thus define the universe of assessments that can be conceived.

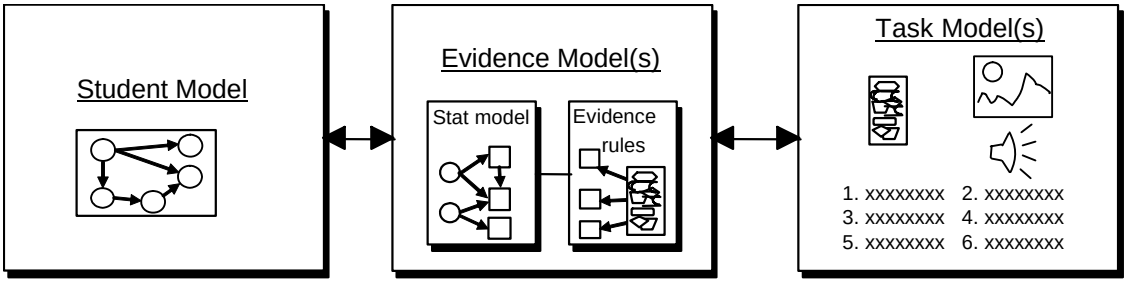

Figure 1. The Central Models of the Conceptual Assessment Framework (CAF)

Student Model

Scores from assessments synthesize evidence from students' performances, in terms of variables organized around KSAs (Knowledge, Skills and Abilities). These are student-model variables. Each variable corresponds to some aspect of Knowledge, skill, ability, proficiency, etc., presumed to drive probabilities of observable response. Psychometric models such as classical test theory, item response theory, and cognitive-diagnosis models use probability-based methods to ground inferences about students and characterize their precision. Especially in applications requiring multivariate models, designers will find it useful to use technology-based tools to construct modular components of the required student models and assemble them for diagnostic assessment and intelligent tutoring systems, such as van Lehn's ANDES physics tutor (Conati, Gertner, & VanLehn, 2002). These re-usable student-model pieces are combined with the measurement-model pieces described below, allowing assessment designers to create tasks that support detailed feedback without having to create complicated measurement models anew for each task.

Reusability of measurement models is of particular importance for not only tasks requiring multivariate models, but also those with multiple scores, dependencies, or modeled relationships between task features and measurement parameters (Rupp, 2002). The psychometric implications of these relationships usually lie outside the expertise of domain specialists who create tasks. However, task authors can be provided a library of preconfigured measurement fragments around which to write any number of unique tasks (Almond, Steinberg, & Mislevy, 2002). When these kinds of supports are available, designers can link task models to students and evidence model fragments, so that the designers are not constrained to unidimensional or "overall proficiency" arguments in domain modeling.

Evidence Model

The evidence model provides a mechanism for connecting observable student behaviors to student-model variables. The evidence model has two components, herein called the *evaluative* and *statistical submodels* but referred to elsewhere by other names, including *evidence extraction* and *evidence synthesis* (e.g., Williamson et al., 2006).

The *evaluative submodel* involves extracting the features that are relevant to a given student-model variable from observable behavior (or some work product), and then judging those features in terms of the evidence they provide about the KSAs of interest. In simple situations, this may just be correctness. In tasks where examinees produce more complex performances, this may be efficiency of problem-solving steps, appropriateness of cohesiveness devices in an essay, or time-on-target in a training exercise. This extraction-and-judgment operation can be performed by machine, by human rater, or by some combination of the two. In the case of a multiple-choice test, the operation is straightforward because the features are represented by the penciled marks on a scannable answer sheet, and judging the correctness of each mark is governed by the match between the mark's position and an answer key. For a constructed response, the rater (human or automated) must locate relevant features and then judge the quality of each feature. Again, knowing what technologies will be available to support evaluation determines the range of performances a designer will note in Domain Analysis, and it determines the range of possibilities for capturing and evaluating evidence that will be available for constructing assessment arguments in Domain Modeling. Williamson, Mislevy, and Bejar (2006)

provide in-depth discussion of automated methods for evaluating complex performances, from the perspective of ECD.

Various commercially-used technology tools exist for feature extraction and judgment. For example, the Online Scoring Network (OSN; Odendahl, 1999) represents a class of tool that displays the student's constructed response (or a digitized image of it) on a computer screen for the human rater to process. Such tools must be able to aid the human rater in extracting and judging features relevant to the assessment argument and avoiding ones tangential to it. For example, OSN allows the rater to highlight and annotate portions of a digitized response as a means of organizing relevant evidence. A compatible kind of technological support for human scoring provides underlying communication and data-base capabilities that allow scorers at different locations to evaluate work products, and that allow assessment designers to monitor, analyze, and improve raters' performances (Whalen & Bejar, 1998).

Other technology tools are capable both of automatically extracting relevant features from constructed-responses and judging their adequacy. Examples of tools for evaluating essay responses include *e-rater* (Burstein, 2003), the *Intelligent Essay Assessor* (Landauer, Laham, & Foltz, 2000), and *Project Essay Grade* (Page, 2003). The features such tools extract and judge, however, must also be aligned with the ones assessment designers intended as evidence for informing standing on student-model variables. A common issue with such tools has been the extent to which they include easily computed but substantively questionable features that, because of their correlation with student-model variables, can stand in for more substantive but difficult-to-extract features like audience awareness (Bennett, 2006).

The *statistical submodel* is the second component in the evidence-model. The statistical submodel provides a mechanism for combining the judged features in a principled manner, in terms of a "score" or value for one or more student model variables. Ideally this synthesis would include an indication of how much evidence should be associated with each resulting score, as provided by item parameters in item response theory models or weights in summed scores. Dependencies among the several features of a complex performance can be difficult to model. However, as with student models, technology supports make it possible to construct appropriate statistical models from pre-constructed modules in computer based assessment (Rupp, 2002), so that a task designer can build unique tasks, guided conceptually by design patterns, around configurable evidence structures. In this way, technology supports based on an ECD or similar

unified assessment framework can improve the designer's work as to both conception and implementation.

The statistical submodel may additionally incorporate multiple features of a complex performance, to produce an intermediate item-level score. This is common in automated essay scoring. Regardless of whether the aggregation is within items, across items, or both, the aggregation should be done in a manner that is consistent with the assessment argument. For example, the use by some automated scoring programs of such brute-empirical methods as step-wise regression undermines the assessment argument because the features selected and weights assigned may diverge from one task and examinee sample to the next, as well as from the judgments of writing experts (Bennett, 2006; Bennett & Ben-Simon, 2006).

Task Model

During domain analysis the designer identifies potential tasks to elicit the relevant student variables. A task model is a more detailed structure that includes information about how the information it elicits is related to other components of the assessment, and it serves as the blueprint for instantiating actual tasks to be presented to the student. Item forms and item models, mentioned as forms of domain modeling, can serve as the basis for task models by augmenting them with *task-model variables*. Alternatively, for tasks that call for complex responses, a task template (Riconscente, Mislevy, Hamel, 2005) is a means of reaching the definition of a task model. In either case, a task model is a schema with variables that are needed to make the task model come alive, so to speak.

Figure 2 shows task-model variables in context. The left of Figure 2 shows the flow of information at assessment time. A student is presented a task and then produces a work product such as a written response, a graphical design, a mathematical formula, or a response log. The work product is then analyzed to extract response variables (evidence extraction) and possibly to aggregate them (evidence synthesis). The result is a set of observable variables that characterizes the work product in a manner consistent with the statistical model that updates the student model variable(s) via a suitable measurement model. Figure 2 also shows some possible task-model variables. The variable shell refers to a template that contains details about the task as it appears to the student. The content variables indicate key features of stimulus material, domain focus, knowledge requirements, and the like, which can be used to select tasks or

assemble them into tests. Figure 2 assumes that a task model is the basis for producing instances (i.e., particular tasks) and, therefore a subset of the task-model variables are designated as instantiation variables. Among the instantiation variables, some are designated as the basis of the "scoring key" of specific instances. If automated scoring is in effect, these variables are data for a scoring engine (Braun, Bejar, & Williamson, 2006). Alternatively, if scoring is done by judges, these variables inform the scoring rubric for each specific instance of the task model. Figure 2 also assumes that the domain analysis and modeling have been sufficiently thorough that a *difficulty model* is available to estimate or impute the difficulty of specific task-model instances (e.g., Enright & Sheehan, 2002). Finally, Figure 2 assumes that the language in which a task is rendered is a variable (see Higgins, Futagi, Deane, 2005). A relational database can be designed to hold all the information for tasks. The challenge in any particular project is to enable queries that are relevant to assessment design and to display the results in a form amenable to design reasoning, for example in the form of a task design "wizard" (e.g., Hamel & Schank, 2006).

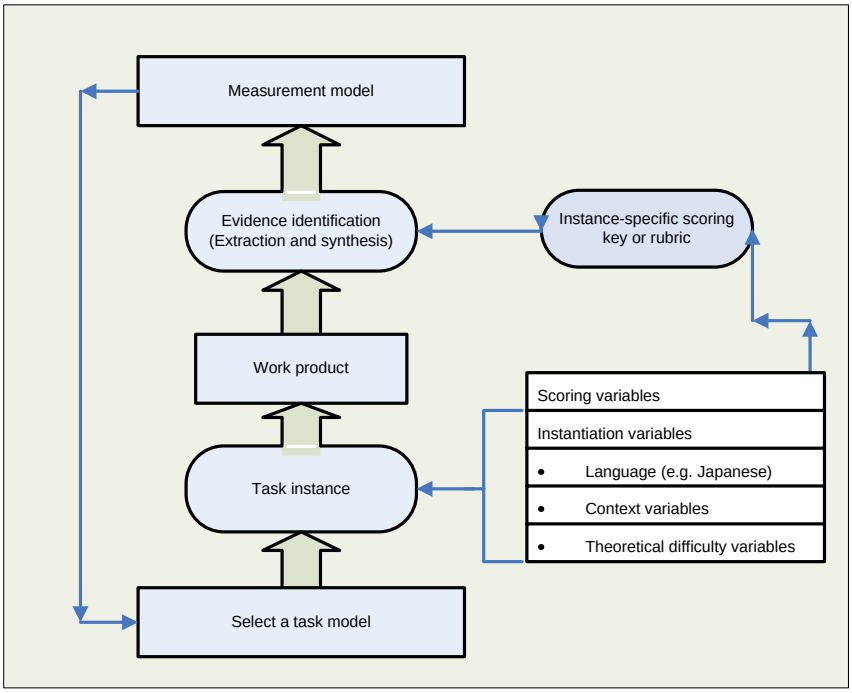

Figure 2. Schematic Showing Two Roles of Task Models Variables

VI. Supports for Implementation

Task Authoring Support Tools

Task authoring tools help developers create tests more efficiently. Many such tools exist, especially for authoring multiple-choice and other selected-response items, and then banking them so that they can be easily retrieved and then assembling them into tests. Such tools as QuestionMark Perception incorporate item templates that facilitate paper as well as on-screen formatting, permitting a full preview of the layout as the item will appear to the examinee. But in addition to improving efficiency, those tools should ideally support the assessment argument by channeling the developer toward some types of implementation and away from others. Tools that in some way incorporate considerations found important in Domain Analysis can be particularly helpful in this regard.

An instructive example is SourceFinder (Passoneau, Hemat, Plante, & Sheehan, 2002; Sheehan, Kostin, Futagi, Hemat, & Zuckerman, 2006), which assists test developers with one particular aspect of the item authoring process. SourceFinder searches online repositories (e.g., databases of literary and scientific journals) to help the test developer more quickly locate appropriate passages for graduate admissions-level reading comprehension questions. The tool uses natural-language processing techniques to characterize each text selection in terms of critical features. The critical features were chosen because of their relationship to the assessment argument, in particular, and their potential to influence passage difficulty in construct-relevant and irrelevant ways. For example, “level of argumentation” features ensure that text selections contain conflict, divergent ideas, uncertainty about outcomes, or other characteristics that can serve as the basis for questions that call upon the targeted verbal reasoning construct. Other features, such as detecting specialized jargon, center on reducing irrelevant variance. Echoing the theme of modularity, SourceFinder allows features to be added or removed as new natural language processing methods emerge or as the assessment argument changes to accommodate, for example, a different test purpose or population.

A second example of a tool that incorporates considerations found important in domain analysis is the Mathematics Test Creation Assistant (TCA) (Singley & Bennett, 2002). The TCA

helps task designers in two ways to make task authoring more efficient and more strongly connected to the research carried out in domain analysis and the argument construction carried out in domain modeling. First, TCA allows the test developer to create “item models,” abstract descriptions that are more general than specific test items, essentially constrained versions of task models. Working at this more general level of abstraction brings the test developer conceptually closer to the categories that comprise the domain analysis than would be possible by crafting each test question individually. Then task designers can author many instances of tasks from the item models.

Creating individual tasks can further be supported with technology assistance. An example of this type is Katz’s (1995) FRADSS (Free-Response Authoring, Delivery, and Scoring System) task-authoring system, which permits test developers to construct computer-based items from "objects." Each object brings with it capabilities that enable the item to behave in certain ways (e.g., present an animation), or the examinee to act upon it (e.g., draw a line, shade a portion of a figure, move figures). The developer can create items from various combinations of objects, interact with them as would the examinee, revise them in real time, assemble a test, and deliver it in pilot form.

Automated Task Generation

Within the framework described above, task generation refers to instantiating an actual task as a function of the variables that comprise the task model. Generation can be fully automated or partially “manual” depending on the complexity of the stem or stimulus material. In the manual case *task specifications* are instantiated rather than ready- to-deliver tasks. For example, assessment of architectural expertise by means of design problem solving is an instance requiring task specifications that are then used by subject-matter experts to finalize a task (Bejar, 2002). Skills such as reading comprehension call for a variety of task models, some of which can be instantiated automatically. For example, consider a format that asks the student to summarize a paragraph where difficulty is to a large extent a function of the attributes of the text to be summarized. Automating the instantiations of such a task requires a detailed text model that describes by means of a set of variables the attributes of the text, such as genre, lexical, syntactic, and other relevant textual attributes (Deane, Sheehan, Sabatini, Futagi, Kostin, 2006). In addition, a statistical model that predicts the likely difficulty of text supplies parameter values to

the evidence model. The predicted difficulty for each potential text can be stored along with the text to facilitate retrieval. Retrieving candidate texts of a given difficulty from a corpus becomes a simple database query (Sheehan, Kostin, Futagi, Hemat, & Zuckerman, 2006).

In domains such as mathematics (e.g., Bejar, et al., 2003) or fluid intelligence (Embretson, 1999; Hornke, 2002; Newstead, Bradon, Handley, Dennis, & Evans, 2006) automated instantiation of task models has reached operational status. Task models, item structures, or their equivalents are recast as instantiation variables (see Figure 2) and become part of the task model. Before such recasting can happen, however, the task model must be authored. Math TCA, mentioned above, has also been used for authoring item models (see Graf, Peterson, Steffen, & Lawless, 2005).

VII. Supports for Delivery

Technology-rich delivery environments such as simulations are complex and expensive to create and to score. Complexity means that there are many opportunities for assessment designers to inadvertently do a good job measuring unintended KSAs and a bad job measuring the intended ones. The expense associated with creation and scoring means that recovering from bad design decisions will almost certainly be very costly and very possibly infeasible. Thus, the delivery environment should attempt from inception to reinforce the assessment argument by minimizing construct-irrelevant variance and maximizing construct-relevant variance. Information about relevant knowledge and ways to obtain it is best addressed early on, in Domain Analysis and Domain Modeling (Luecht, 2002). The System of Intelligent Evaluation using Tests (SIETTE) used in Europe in several projects incorporates this philosophy (Conejo, et al., 2004); in addition to managing web-based adaptive testing, SIETTE supports item models and multilingual tests.

The delivery platform used in the NAEP Technology-Rich Environments project offers an example of technology support for delivery that addresses considerations from domain analysis and domain modeling (Bennett, Jenkins, Persky, & Weiss, 2003). In this project, 8th-grade students were asked to use the computer for problem solving in a science context. The task involves conducting simulated experiments to discover the relationship between a set of quantities (the payload mass carried by a helium gas balloon and the altitude to which it can rise in the atmosphere). Student behavior in this environment provides evidence for several student-

model variables, including a general “problem-solving with technology” variable, and two more specific ones, “scientific inquiry” and “computer skills.”

To minimize irrelevant variance associated with learning the environment, the tools employed for conducting experiments are explained in a brief tutorial and, also, utilize such common software conventions as dialog boxes and wizards. Further, the tools are organized around a representation of the experimental process designed to reinforce what students are expected to do substantively (i.e., design experiment, run experiment, interpret results).

Evidence for the student model variables is gathered from two sources, both of which are intended to contribute to construct-relevant variance. One key source of evidence is the adequacy and completeness of the written description of the relationship between payload mass and altitude that the student provides. A second source of evidence is how the student uses the tools to arrive at that relationship. Tool use is relevant only because the tools, by design, provide conceptual and data structures to support the assessment argument. For example, the student’s use of the “design experiment” and “run experiment” tools indicates whether the student has executed enough experiments—and covered the range of payload masses sufficiently—to support a defensible conclusion about the relationship between payload mass and altitude. Similarly, his or her use of the “interpret experiment” tools indicates whether a table or graph was created that even includes the two variables relevant to solving the problem.

VIII. Conclusion

Designing an assessment entails many carefully considered steps, or layers. At each layer, technology supports can aid the thinking and building of the processes and elements that are needed to embody the assessment argument. They can make particular tasks more efficient, as well as open up new possibilities for tasks that were previously too expensive, difficult, or not even considered. However, the focus should not be on the technology per se, but rather the ways in which it can support the process of designing an assessment so that it fully embodies the assessment argument. Thinking about assessment and technology supports from the perspective of assessment arguments, layers, and knowledge representations provides a method for understanding and evaluating the roles of current technology supports for assessment design as it exists today as well as in the future.

Bibliography

Alexander, C., Ishikawa, S., Silverstein, M., et al. (1977). A Pattern Language: Towns, Buildings, Construction. New York: Oxford University Press.

Almond, R. G., Steinberg, L. S., & Mislevy, R. J. (2002). Enhancing the design and delivery of assessment systems: A four-process architecture. *Journal of Technology, Learning, and Assessment, 1*(5).

Bejar, I. I. (2002). Generative testing: From conception to implementation. In S. H. Irvine & P. C. Kyllonen (Eds.), Item generation for test development (pp.199-218). Mahwah, New Jersey: Erlbaum.

Bejar, I. I., Lawless, R. R., Morley, M. E., Wagner, M. E., Bennett, R. E. & Revuelta, J. (2003). A feasibility study of on-the-fly adaptive testing. *Journal of Technology, Learning and Assessment, 2*(3) 4.29.

Bennett, R. E. (2006). Moving the field forward: Some thoughts on validity and automated scoring. In D. M. Williamson, R. J. Mislevy, & I. I. Bejar (Eds.), *Automated scoring of complex tasks in computer-based testing*, pp. 403-412. Hillsdale, NJ: Erlbaum.

Bennett R. E., & Ben-Simon, A. (2006). Toward theoretically meaningful automated essay scoring. Jerusalem, Israel: National Institute for Testing and Evaluation (NITE). Retrieved January 6, 2007 from: http://clickit.ort.org.il/files/upl/224004906/934404432.pdf .

Bennett, R.E., Jenkins, F., Persky, H., & Weiss, A. (2003). Assessing complex problem-solving performances. *Assessment in Education, 10*, 347-359.

Braun, H., Bejar, I. I., & Williamson, D. M. (2006). Rule based methods and mental modeling. In D. M. Williamson, R. J. Mislevy, & I. I. Bejar (Eds.), Automated scoring of complex tasks in computer based testing (pp. 83-122). Mahwah, NJ: Lawrence Erlbaum.

Burstein, J. (2003). The e-rater® scoring engine: Automated essay scoring with natural language processing. In M. D. Shermis & J. C. Burstein (Eds.), *Automated essay scoring: A cross-disciplinary perspective*. Mahwah, NJ: Erlbaum.

Conati, C., Gertner, A., & VanLehn, K. (2002). Using Bayesian networks to manage uncertainly in student modeling. *User Modeling & User-Adapted Interaction*, 12(4), 371-417.

Conejo, R., Guzmán, E., Millán, E., Trella, M., Pérez-De-La-Cruz, J. L., & Ríos, A. (2004). SIETTE *International Journal of Artificial Intelligence in Education*: A Web-Based Tool for Adaptive Testing. *International Journal of Artificial Intelligence in Education 14*, 29-61.

Davidson, F., & Lynch, B.K. (2001). *Testcraft: A teacher's guide to writing and using language test specifications*. New Haven: Yale University Press.

Deane, P., Sheehan, K. M., Sabatini, J., Futagi, Y., & Kostin, I. (2006). Differences in text structure and its implications for assessment of struggling readers. *Scientific studies of reading 10* (3), 257 – 275.

Embretson, S. E. (1998). A cognitive design system approach to generating valid tests: Application to abstract reasoning. *Psychological Methods, 3,* 380-396.

Embretson, S.E. (1999). Generating items during testing: Psychometric issues and models. *Psychometrika, 64*, 407-433.

Enright, M. K., & Sheehan, K. (2002). Modeling the difficulty of quantitative reasoning items: Implications for item generation. In S. H. Irvine & P. Kyllonen (Eds.), Item generation for test development (pp. 129-158). Mahwah, NJ: Lawrence Erlbaum Associates.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995). *Design Patterns: Elements of Reunsable Object-Oriented Software*. Upper Saddle River, NJ: Pearson.

Graf, E. A., Peterson, S., Steffen, M., Lawless, R. (2005). *Psychometric and cognitive analysis as a basis for the revision of quantitative item models* (No. RR-05-25). Princeton, NJ: Educational Testing Service.

Hamel, L., & Schank, P. (2006). *A Wizard for PADI assessment design (PADI Technical Report 11).* Menlo Park, CA: SRI International.

Higgins, D., Futagi, Y., & Deane, P. (2005). Multilingual generalization of the ModelCreator software for math item generation (RR-05-02) ETS Research Report. Princeton, NJ: Educational Testing Service.

Hively, W., Patterson, H.L., & Page, S.H. (1968). A "universe-defined" system of arithmetic achievement tests. *Journal of Educational Measurement, 5*, 275-290.

Hornke, L. F. (2002). Item generation models for higher cognitive functions. In S. H. Irvine & P. C. Kyllonen (Eds.), *Item generation for test development* (pp. 159-178). Mahwah, New Jersey: Earlbaum.

Kane, M. T. (1992). An argument-based approach to validity. *Psychological Bulletin, 112* (3), 527-535.

Katz, I. R. (1995). FRADS: A system for facilitating rapid prototyping by end users. In Y. Anzai & K. Ogawa (Eds.), *Proceedings of the Sixth Annual International Conference on Human-Computer Interaction.* Amsterdam: Elsevier Science Publishers.

Kurland, L., Gertner, A., Bartee, T., Chisholm, M., & McQuade, S. (2005) *Using cognitive task analysis and eye tracking to understand imagery analysis.* Mitre Corporation Technical Report #05-1365. Bedford, MA: Mitre Corporation.

Landauer, T.K., Laham, D., & Foltz, P.W. (2000). The Intelligent Essay Assessor. *IEEE Intelligent Systems, 15,* 27-31.

Luecht, R. M. (2002). From design to delivery: Engineering the mass production of complex performance assessments. Paper presented at the Annual Meeting of the National Council on Measurement in Education, New Orleans, LA.

Messick, S. (1994). The interplay of evidence and consequences in the validation of performance assessments. *Educational Researcher, 23*(2), 13-23.

Mislevy, R.J., & Riconscente, M. (2006). Evidence-centered assessment design: Layers, concepts, and terminology. In S. Downing & T. Haladyna (Eds.), *Handbook of Test Development.* Mahwah, NJ: Erlbaum.

Mislevy, R., Steinberg, L. S., Almond, R. G. (2003). On the structure of educational assessments. *Measurement: Interdisciplinary Research and Perspectives, 1*(1), 3-62.

Newstead, S. E., Bradon, P., Handley, S. J., Dennis, I., & Evans, J. S. B. T. (2006). Predicting the difficulty of complex logical reasoning problems. *Thinking & Reasoning, 12*(1), 62-90.

Odendahl, N. (1999, April). *Online delivery and scoring of constructed-response assessments*. Paper presented at the annual meeting of the American Educational Research Association, Montreal, Quebec.

Page, E. B. (2003). Project Essay Grade (PEG). In M. D. Shermis & J. C. Burstein (Eds.), *Automated essay scoring: A cross-disciplinary perspective*. Mahwah, NJ: Erlbaum.

Passoneau, R., Hemat, L., Plante, J., & Sheehan, K. M. (2002). Electronic sources as input to GRE reading comprehension item development: Sourcefinder prototype evaluation (RR-02-12). Princeton, NJ: Educational Testing Service. Retrieved November 25, 2006 from: http://www.ets.org.

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

Rupp, A.A. (2002). Feature selection for choosing and assembling measurement models: a building-block-based organization. *International Journal of Testing, 2,* 311-360.

Sheehan, K. M., Kostin, I., Futagi, Y., Hemat, R., & Zuckerman, D. (2006). *Inside SourceFinder: Predicting the acceptability status of candidate reading comprehension source documents* (RR-06-24). Princeton, NJ: Educational Testing Service. Retrieved November 25, 2006 from: http://www.ets.org.

Singley, M. K., & Bennett, R. E. (2002). Item generation and beyond: Applications of schema theory to mathematics assessment. In S. Irvine & P. Kyllonen (Eds.), Item generation for test development (pp.361-384). Hillsdale, NJ: Erlbaum.

Shute, V.J., Torreano, L. A., & Willis, R. E. (2000). DNA: Toward an automated knowledge elicitation and organization tool. In S. Lajoie (Ed.). *Computers as cognitive tools: No more walls, II*. Mahwah, NJ: Erlbaum.

Van der Linden, W.J. (2005). *Linear models for optimal test design*. New York: Springer-Verlag.

Whalen, S. J., & Bejar, I. I. (1998). Relational databases in assessment: An application to online scoring. *Journal of Educational Computing Research, 18*, 1-13.

Williams, K. E. (2000). An automated aid for the conduct of a detailed cognitive task analysis for modeling human computer interaction performance. In .R. Schraagen, S. F. Chipman and V. L. Shalin (Eds), *Special series on cognitive task analysis*, pp.165-180. Hillsdale, N.J.: Erlbaum.

Williamson, D. M., Mislevy, R. J., Bejar, I. I. (Eds.) (2006). *Automated scoring of complex tasks in computer-based testing.* Mahwah, NJ: Erlbaum.

Zachary, W. W.; Ryder, J. M., & Hicinbothom, J. H. (2000). Building cognitive task analyses and models of a decision-making team in a complex real-time environment. In Chipman, Shalin & Schraagen, (Eds). *Cognitive task analysis*, pp. 365-384. Mahwah, N.J.: Erlbaum.

Further Reading

American Association for the Advancement of Science. (2001). *Atlas of Science Literacy*. Washington D.C.

Baker, E. L. (2002). Design of automated authoring systems for tests. *Proceedings of Technology and assessment: Thinking ahead proceedings from a workshop*." (pp. 79-89). Published of Collection: National Research Council, Board on Testing and Assessment, Center for Education, Division of Behavioral and Social Sciences and Education.

Messick, S. (1989). Validity. In R. L. Linn (Ed.). Educational Measurement (3rd ed., pp. 13-103). New York: American Council on Education/MacMillan.

Mislevy, R., & Haertel, G. (2006). *Implications of Evidence-Centered Design for Educational Testing (Draft PADI Technical Report 17).* Menlo Park, CA: SRI International.

Mislevy, R.J., & Haertel, G.D. (in press) Implications of evidence-centered design for educational assessment. *Educational Measurement: Issues and Practice.*

Simon, H.A. (2001). Complex systems: The interplay of organizations and markets in contemporary society. *Computational & Mathematical Organization Theory, 7*(2), 79-85.

VanLehn, K. & Niu, Z. (2001). Bayesian student modeling, user interfaces and feedback: A sensitivity analysis. *International Journal of Artificial Intelligence in Education*, 12(2) 154-184.

Williams, K.E., Hultman, E. G., and Graesser, A. (1998). CAT: A tool for eliciting knowledge on how to perform procedures. *Behavior Research Methods, Instruments & Computers, 30* (4), 565-572.

Wilson, M. R. (2005). *Constructing measures: An item response modeling approach.* Mahwah, NJ: Erlbaum.