ETS

Research Report

Model Analysis and Model Creation: Capturing the Task-Model Structure of Quantitative Item Domains

Paul Deane

Edith Aurora Graf

Derrick Higgins

Yoko Futagi

René Lawless

Research & Development

May 2006
RR-06-11

Model Analysis and Model Creation: Capturing the Task-Model Structure of Quantitative Item Domains

Paul Deane, Edith Aurora Graf, Derrick Higgins, Yoko Futagi, and René Lawless
ETS, Princeton, NJ

May 2006

Abstract

This study focuses on the relationship between item modeling and evidence-centered design (ECD); it considers how an appropriately generalized item modeling software tool can support systematic identification and exploitation of task-model variables, and then examines the feasibility of this goal, using linear-equation items as a test case. The first half of the study examines task-model structures for linear equations and their relevance to item difficulty within ECD. The second half of the study presents prototype software, a Model Creator system for pure math items, designed to partially automate the creation of variant item models reflecting different combinations of task-model variables. The prototype is applied to linear equations but is designed to generalize over a range of pure mathematical content types.

Key words: Item modeling, item generation, evidence-centered design, ECD, task modeling, Model Creator, mathematics cognition, mathematics assessment, item difficulty

Acknowledgments

Thanks are due to Kathleen Sheehan, for meeting on several occasions and providing us with valuable suggestions regarding the difficulty impact analysis, to Sandip Sinharay for his feedback on the statistical analysis, and to Shelby Haberman for his advice. Thanks also to Isaac Bejar, Valerie Shute, Kathleen Sheehan, and Dan Eignor, for their extensive comments and suggestions on an earlier draft of this report. We appreciate the advice of these contributors, and they are not responsible for any errors contained herein. Finally, we thank Stephen Peterson for his feedback and Kim Fryer for coordinating the copy editing.

Table of Contents

List of Tables

List of Figures

Page

Item Modeling and Evidence-Centered Design

Item Modeling

In recent years, ETS has focused considerable effort on the creation of systems to support automatic item generation, particularly in the form generally referred to as *item modeling* (see Bejar, 1993, 1996, 2002; Bejar, Lawless, Morley, Wagner, Bennett, & Revuelta, 2002; Bejar & Yocom, 1991). Item modeling is closely related to automatic item generation (AIG; Kyllonen, 2002), such that item models specify the schemas, or conceptual frameworks, that inform item generation (Singley & Bennett, 2002) and, in its relation to AIG, it can serve a number of goals: for example, supporting adaptive testing, improving test security, and speeding or partially automating item creation. Item modeling can be distinguished from other item-generation approaches by its emphasis on creating items whose psychometric properties are predictable. Item models may be designed to generate *variants* (i.e., items that vary systematically along a variety of dimensions), or they may be designed to generate *isomorphs,* that is, sets of items thought to be mutually interchangeable in terms of content and item operating characteristics (Bejar, 2002). Isomorphic item models are a special case in which variations in the task[1] defined by the item model are limited to those thought to have minimal impact on their psychometric properties (e.g., on item characteristics). The goal of isomorphic item models is to create families of items whose surface features differ but which measure the same underlying construct and have similar psychometric properties.

The primary vehicle for mathematics item modeling at ETS is a tool termed the Math Test Creation Assistant, or Math TCA (Singley & Bennett, 2002). The architecture of Math TCA is based upon three fundamental assumptions: first, that the presentation structure of items can be modeled with simple templates; second, that quantitative items can be modeled using a mathematical constraint language; and third, that item models can be grouped into families, whereby an item model can be a submodel or child of another, along the lines of schema theory (Marshall, 1995; Mayer, 1982b). No explicit provision is made for relating item models or their generated instances (i.e., items) to an analysis of task-model variables.

The following item model, covering a range of linear and simple rational equation items, illustrates how Math TCA typically is applied to item modeling. An item model contains three parts: a presentation template, variable definitions, and a set of mathematical constraints. In this example, the presentation template for the item model is shown in the first column of Figure 1.

Model presentation template	Variable definitions	Constraints
What value of *$V* satisfies the equation: *$V – N = N – $V?* *Key* *Distractor1* *Distractor2* *Distractor3* *Distractor4*	*$V:* string variable[2] with values "*n,*" "*x,*" "*y,*" or "*z.*" *N:* independent integer variable between 1 and 25. *Key, Distractor1, Distractor2, Distractor3, Distractor4:* dependent integer variables.	*Key - N = N - Key* *Distractor1: N + 1* *Distractor2: 0* *Distractor3: -(N + 1)* *Distractor4: N - 1*

***Figure 1.* Presentation template, variables, and constraints for a simple linear-equation item model.**

Note. For purpose of demonstration, variables are shown in italics; string variables are preceded by "$."

The key to this structure is that it consists of a single document with *variable insertion slots* into which the values of variables can be substituted. That is, the presentation side of Math TCA uses a template-based system.

Its mathematical constraint system is the core of Math TCA. A set of variables can be defined and their relationships specified; a built-in constraint engine solves the system of constraints and yields a set of possible solutions as output; and instances of the model are generated by substituting solution values into the variable insertion slots in the presentation template. For instance, the model with the presentation template shown in the first column of Figure 1 is defined as satisfying the system of constraints shown in the third column.

The variable definitions portion of this model includes at a minimum all variables that are specified in the model presentation template; it defines the variables used in the key and the distractors, as well as the variables used in the stem of the item. The definition for each variable specifies the type of values that may be assumed (integer, decimal, fraction, or string). For numeric variables, it specifies whether the variable is independent or dependent. For independent variables, inclusive lower and upper bounds and an increment are included in the definition.

Equations that constrain the possible combinations of values that will yield an acceptable solution are defined in the constraint portion of the model. Constraints are used to define dependent variables or to restrict the values that independent variables may assume. Having defined a model such as that shown in Figure 1, Math TCA allows new models, referred to as *submodels*, to be created as children. Typically, submodels are altered with respect to the parent. Changes may be made to the presentation template, the variable definitions, or the constraints.

Evidence-Centered Design

Another recent development in assessment theory is *evidence-centered design*, or *ECD* (Mislevy, Almond, & Lukas, 2003; Mislevy & Riconscente, 2005; Mislevy, Steinberg, & Almond, 2003). ECD is a model of how to ground assessment development by explicitly linking the form and content of test items to their evidentiary role with regard to the skill the test developer wants to measure. The key to the ECD perspective is that it requires a systematic analysis of the construct to be measured in an assessment, and it provides a framework (the *conceptual assessment framework*) that allows systematic distinctions to be made among different aspects of the assessment development process. The term *model* is used within ECD for almost all component elements of the conceptual assessment framework. Models in this sense should not be confused with item models as described above, as they function at a much higher level of abstraction and capture the relationships among the various aspects of the assessment process, whereas item models are much more specific and are concerned with relatively narrow variations within specific single content areas and task types. The most relevant ECD models from an item modeling perspective include:

- The *student model*, which specifies which aspects of student knowledge and capabilities are to be assessed. The student model is defined in terms of latent variables that approximate selected aspects of student skills and knowledge in some domain, and it is related to actual performance by constructing arguments in which particular tasks are interpreted as providing evidence for particular knowledge, skills, or attributes. The student model is thus an abstract description of student attributes that is fleshed out by specifying evidence and task models. It is sometimes useful to distinguish between a general *proficiency model*, which specifies the predictive variables and their relationships, and a specific *student model* that contains appropriate values for specific students. In what follows we will use the term student model, as we will not have occasion to distinguish the general model of student proficiency from specific student models.

- The *evidence model*, which specifies why or how displayed behaviors or performances in a given task situation constitute evidence for student-model variables. The evidence model contains two subparts: the *evaluative submodel*, which deals with extracting salient features from the student work product, and the *statistical submodel*, which indicates how these features depend probabilistically on student-model variables. Construction of the evidence model depends critically upon justifying an evidentiary relationship between student-model variables and evidence model features.
- The *task model*, which provides a framework that specifies tasks or situations within which evidence model features can be observed. It includes specifications of the assessment situation (what materials examinees will see; what they are expected to say or do) and the work product (the form in which their performance is captured). Task-model variables play multiple roles. On the one hand, they may implicitly or explicitly define conditions under which the evidence model will yield valid inferences about student-model variables; on the other hand, they directly specify parameters of task design and thus affect task construction, test assembly, and presentation.

In addition, ECD specifies further models that govern test delivery, for example, the assembly model, the presentation system model, and the delivery system model. ECD, therefore, can be viewed as a method for organizing the variety of activities involved in assessment design and administration into a single coherent process focused on maintaining construct relevance and validity.

Item Modeling and ECD: The Task-Modeling Problem

The prime focus of this study is the relationship between item modeling and ECD; in particular, we consider how an appropriately generalized item-modeling software tool can be used to support systematic identification and exploitation of task-model variables. Mislevy, Almond, and Lukas (2003) characterized task models as follows:

> A task model does not represent a single task, but rather a family of potential tasks waiting to be written. Tasks are made from task models by filling in the specification made by the task model, that is, finding or authoring presentation material and setting the

> value of task-model variables. A typical assessment will have many task models representing different families of tasks. A family of tasks produced from a task model is not necessarily meant to be psychometrically interchangeable. That is, they are not item clones for producing alternate forms, but are a structure for understanding and controlling the evidential variation, often systematically manipulating the evidential value (and statistical parameters) of the item in question. (p. 12)

As noted above, most existing work on item models does not focus on the possibility of using item models as task models but rather on the use of item modeling to create isomorphs with similar psychometric properties. Thus, there is considerable work on the statistical equivalence of variants within a single item model and on the development of methods to account for their psychometric variance using extensions of item response theory (IRT) techniques (Johnson & Sinharay, 2003; Sinharay & Johnson, 2005; Sinharay, Johnson, & Williamson, 2003). Even at this relatively fine grain size, however, a significant proportion of item models turn out to have a high proportion of instances whose IRT parameters are significantly at variance from the mean for the model, though usually this variation turns out to be due to features of the task not controlled for in the item model (Graf, Peterson, Steffen, & Lawless, 2005; Meisner, Luecht, & Reckase, 1993). The problem of accounting for item variation at a task-model level is intrinsically more complex, involving a much wider range of variation and generalizations over items at a significantly higher level of abstraction. It is thus an open question whether software tools can be developed that will enable at least partial automation of task modeling (as much as existing item-generation software enables the variation of existing items). The main purpose of this study is to explore strategies for automating task modeling and to present a prototype software system that supports task modeling for a significant class of quantitative item types.

Task modeling, as outlined in Mislevy, Almond, and Lukas (2003), assumes that separate task models will be developed for each major task in an assessment. Tasks differing significantly in content or in presentation format are unlikely to be comparable with respect to evidence or student-model variables and should be handled separately. Task modeling thus can be conceived of as the problem of modeling structural *variation among items that test the same or very similar skills*. Such items vary both in form and content, on the one hand, and in psychometric properties, on the other, and in most content area domains very little is known about the factors that determine whether or how a change in form or content will impact item operating characteristics. The key point is that apparently small form and content changes can have a large influence on the psychometric properties of the instances, and it can be difficult to predict which changes will have an effect. There are some studies of the features that determine difficulty within some verbal item types (see Sheehan & Kostin, 2006) and a number of studies of the variations in item difficulty for word problems, (e.g., Enright, Morley, & Sheehan, 2002; Goldin & McClintock, 1984; Sebrechts, Enright, Bennett, & Martin., 1996; Riley, Greeno, & Heller, 1983). In addition, there have been a number of studies of the difficulty of perception, abstract reasoning, and pure math items, including hidden figure items (Bejar & Yocom, 1991), spatial rotation items (Bejar, 1990; Embretson, 1994; Embretson & Waxman, 1989), geometric analogies (Embretson & Schneider, 1989; Whitely & Schneider, 1981), abstract reasoning (Embretson, 1998), arithmetic (Hively, Patterson, & Page, 1968), and proportional reasoning (Harel & Behr, 1989; Harel, Behr, Post, & Lesh, 1991), to name some of the most salient. However, many of these studies are concerned with relatively closed classes of items where a large collection of items can be generated by manipulating a small number of variables. This is problematic for large-scale test development purposes, because the majority of tasks used to assess more complex forms of knowledge and reasoning are themselves complex and allow for variation along a variety of dimensions, making it difficult to exhaustively cover all of the possible forms of variation.

One way to view this task is to conceptualize it as the task of determining the universe of item models required to cover all combinations of the variables in an ECD task model for a content domain. The relationships among the item models should reflect the structure of the domain's task model. Differences among isomorphs from the same item model should reflect purely presentational or otherwise irrelevant variables, such as incidental details, with no mathematical significance that are nonetheless varied within a model, and the relationship between the task and evidence model should be clearly specified. Insofar as this division of labor has been performed correctly, and task-model variables have been stated at the right levels of generality, it should be possible to systematically generate item models that illustrate a wide variety of task-model features. Ideally, the task models would be captured in software, just as item models are currently captured, so that the relationship between specific tasks and the task-model variables can be explicitly accounted for. It is not clear a priori, however, what level of generalization is correct, nor is it necessarily clear which task-model features will play an important role in modeling item operational characteristics. Ideally, task-model variables would play only an indirect role. In an ECD perspective, the evidence-model and student-model variables should account for differences in performance; if task-model variables introduce further variation not correlated with evidence model variables, they are by definition construct irrelevant. However, to the extent that correlations are known to exist, it will be extremely useful to support task modeling and item modeling in ways that enable finer control over item difficulty.

The kind of analysis envisaged here builds upon item-difficulty analyses of the type performed in such studies as Enright and Sheehan (2002). Their analysis considered variation among quantitative items at two levels of abstraction: a specific level, where the task to be solved is more or less completely specified in mathematical terms, and a more general level, where all tasks are analyzed in terms of a common set of features.

At the more specific level, they examined the effect of varying features within what is, conceptually, a single narrowly defined item type and explored the effects of varying specific features. For instance, they examined "round-trip" rate problems where the same distance is covered in different directions at different speeds, and explored the effects of varying the complexity of the numbers used, the real-word context (e.g., distance), and the presence or absence of a variable. In some cases, such as the round-trip rate problems, varying features in this relatively narrow range accounted for more than 90% of the variance in item difficulty, though in other cases, such as probability items, low-level task features were much less predictive of difficulty levels.

At the more generic level, they examined features that could be applied to an entire quantitative item pool, such as general content area (data analysis vs. algebra), specific content (probability, percent, linear inequalities, etc.), skill level required (procedural, higher-order, applied), and item format (quantitative comparison, data sufficiency, etc.), and examined the extent to which item difficulty could be predicted from global features of this sort. The resulting model accounted for 30% of the variance, which while relatively low was consistent with the existence of significant variance within individual item types, though many of the features it applied, such as the judgment of skill level, could only be assigned by applying human judgment to the individual items.

The two levels of Enright and Sheehan's (2002) analysis do not exhaust the possibilities. Between the relatively fine-grained variations at their lower level, and the global variation among task types at their higher level, there are variations within content areas: different types of rate items, different types of probability items, and variations that arguably correspond to the kinds of variation that have to be accounted for at the task-model level. That is, it might make sense to have a single task model that covered all or some large subportion of the class of rate items, or of some other content area, and that would provide a systematic mapping out of the manner in which content-relevant tasks could be structured. Such a task model is rather more abstract than can be modeled in a single Math TCA item model, or possibly even in a family of Math TCA models, but much of the variation among tasks at this level is still systematic and thus potentially subject to automation in a program that integrated item modeling along the lines currently supported by Math TCA, with explicit task modeling over entire classes of content. The problem of modeling difficulty—that is, of identifying which task-model variables affect difficulty, and which of those are relevant to the student model—presupposes some mapping out of the task model first. The more effectively the task model can be automated, the more easily instances can be produced to test hypotheses about item difficulty with respect to the variables to be included in the student model, and thus the problem of providing computational support for task modeling goes hand-in-hand with the development of difficulty modeling and other aspects of ECD.

The basic method of analysis we are discussing can be implemented, in the first instance, by creating a series of item models that illustrate different combinations of task-model variables (although the item models are likely to be examples of, rather than definitions of, specific combinations of values for the relevant task-model variables). The prerequisite for such an analysis is the existence or the ability to generate (from an ECD specification) some large number of items that all fall within a single family in terms of the skill(s) they are designed to test. Such a family might illustrate a class of word problem tasks (such as work problems, taxation and interest problems, or distance/rate/time problems) or it might illustrate a class of pure mathematics items (such as arithmetic sequences, linear equations, or the like). In either case, the word problem class will display dimensions of variation that can be isolated by the systematic comparison of items. After the dimensions of variation have been isolated, they imply a multidimensional classification matrix into which new and existing items can be placed. Such an n-dimensional grid can be viewed as the logical consequence of combining all possible values of task-model variables.

An example of this type of analysis (implemented by defining a set of item models that cover various combinations of task-model variables) was described in Graf and Shute (2004) and Shute, Graf, and Hansen (2005). They present an integration of ECD with item modeling for the mathematical construct of *sequences as patterns*. Mathematical sequences (arithmetic, geometric, and other sequences, such as the Fibonacci numbers, that have a readily identifiable recursive definition) are a significant topic of instruction in early high school mathematics. Graf and Shute prepared both a task model and a set of item models and produced a series of hypotheses about how the elements of the task model would affect difficulty. Their analysis was presented in the form of a tree structure; however, the results can usefully be reconceptualized as a multidimensional grid upon which items can be plotted along several dimensions of variation. In particular, for the category they label *solve problems with recursive sequences*, the following dimensions capture the major dimensions of variation:

Sequence type

1. Arithmetic sequences
2. Geometric sequences
3. Other recursive sequences[3]

Sequence representation

1. Simple list (a, b, c, d …)
2. Chart or table
3. Pictorial pattern
4. Verbal description of a situation that may be modeled by a sequence

Representation of rule for sequence

1. Implicit (not stated or required to be stated)
2. Explicitly stated verbally, using descriptive language
3. Explicitly stated algebraically
    a. Stated as a recursive formula

b. Stated as an explicit formula for the *n*th term

Required response formats

1. Given a sequence, supply additional or missing terms
2. Given a sequence, provide a rule that describes it (see the preceding "Representation of Rule for Sequence," for types of rules that may be required)
3. Given a rule for a sequence, produce the value for the *n*th term
4. Given a rule for a sequence, produce an example sequence that fits the rule

This grid provides what is arguably a (reasonably) complete task-model classification for this construct. It does not, however, exhaust the ways in which item models can vary among themselves. Graf and Shute (2004) drafted specifications for item models that would vary in difficulty from easy to medium to hard for each of the nodes in their classification. To achieve differing levels of difficulty, other aspects of item structure had to be varied systematically following working hypotheses about factors that would affect difficulty. These included:

Properties of the numbers used in a sequence

1. The type of number in the sequence (whole numbers, fractions, decimals)
2. The sign of the numbers in the sequence (positive, negative, both positive and negative)
3. The magnitude of the initial value

Properties of the relationship between numbers in a sequence (common difference, common ratio)

1. The type of number (whole numbers, fractions, decimals) of which the relation is an instance
2. The magnitude of the difference or ratio
3. The complexity of the numerical representation (uncommon fractions vs. simple fractions, large vs. small number of place values)
4. The relationship between the initial value and the recursive rule (e.g., an arithmetic [geometric] sequence where the initial value is equal to common difference [common ratio] has a simpler explicit rule)

It is possible that with a larger set of sample items to draw on, additional properties might emerge, but the grid supplied by this type of analysis provides a structured means not only to develop hypotheses about evidence model variables and item difficulty but also to discipline the mapping from task model to *presentation model* (i.e., how a test item is to be formatted) and to automate the production of item models for specific tasks.

That is, given an analytical, multidimensional grid of this type, one can reasonably assume that an item model that occupies no more than a single cell in the grid should be psychometrically homogeneous. If such item models do not display consistent IRT parameters, one may safely assume that the analysis that produced the grid missed key task-model variables or other confounding factors. This is a minimal expectation. With sufficient data to support the analysis of the interaction among variables, it should be possible to determine, for any cell in the grid, how much evidence it provides for each of the components in the task model. Of course, the number of cells in such a grid is likely to grow very rapidly, and many, perhaps even a majority, may have relatively little value for assessment purposes, in that they represent combinations of task-model variables that yield tasks that are inappropriate, too easy, too hard, or otherwise insufficient to support development of new items for assessment. But it seems clear that creating such a grid of possibilities is foundational for task-model analysis, even when the mapping to a difficulty model and the evaluation of construct-relevance necessary to construct a student model are fairly complex. The problem, in effect, is that there is a need to reconcile two potentially conflicting demands: On the one hand, item generation and item presentation require that information about items be specified fully and explicitly; on the other hand, the task, evidence, and student models require appropriate abstraction to those variables that represent the most significant components of variation among items. Both aspects of analysis—specification of the ECD task specification and specification of item generation and presentation—ideally should be linked and proceed in parallel.

Note that one of the most important aspects of the analytical grid is its potential for providing a strategy for creating new item models. If a user interface is designed to provide control over item modeling at the grid level so that the choices the user makes when they specify an item will correspond to choices of rows and columns in the grid, then the task of designing specific item models for specific assessment tasks will be explicitly aligned with a task-model analysis and potentially with a difficulty model.

The computational work reported later in this paper is designed to take advantage of this fact and to explore what needs to be done to provide computational support for task-model analysis that integrates task modeling with item modeling in an ECD framework. The key point to note is that task modeling requires the ability to create entire families of models that represent particular combinations of task-model variables. In the work reported in Graf and Shute (2004), the task-model variables organized the relationship between the models, but nothing in the models themselves identified which task-model variables they represented or which choices of values for the task-model variables led to the details of the specific model being produced. That is, the task-model analysis was a layer built on top of discrete item models. While the creation of the item models was driven by the task-model analysis, this was a manual process. The Math TCA software does not support the automatic production of item models that reflect particular combinations of task-model variables.

In the rest of this paper, the task of producing an integrated task model that specifies possible variations among item models within a content area will be referred to as *model analysis*, and the task of generating models consistent with a model analysis will be termed *model creation*. There are several reasons why it is worthwhile to take steps to provide computational support for model analysis and model creation:

1. From an ECD perspective, success in these tasks entails the capacity to generate instances that reflect many different combinations of task-model variables.
2. From an item-modeling perspective, explicitly linking generation variables to ECD tasks is desirable in its own right, because (among other things) it can help clarify which potential sources of psychometric variation among instances reflect construct-relevant variation and which do not.

In short, the goal of model creation is to support approaches to item modeling in which the generation variables are explicitly aligned with task-model variables within ECD.

Steps Toward Automating Task-Model Analysis

Math Word Problems

An illustration of the complexities involved in integrating task-model analysis with item generation can be drawn from the domain of distance-rate-time word problems of a specific type analyzed in Deane and Sheehan (2003). Mathematically, this item type (GRE® quantitative comparison word problems involving vehicles) is quite simple, involving a few variations on the base equation $d = rt$ (distance equals rate times time). Despite this, when all possible variations in phrasing, content, and form are considered, there are more than 20 parameters that can be varied independently, ranging from clearly task-relevant variables like the number of parallel equations, the identity of the unknown, or the choice of unit terms to variations that should matter only for presentation purposes, such as whether a specific vehicle type is mentioned, whether active or passive sentences are employed, and similar matters of phrasing. The problem (as yet unsolved) for variations such as this within an item class is how to make the separation between those variables that matter for the evidence model (either directly or indirectly) and those that matter for the task model but do not directly contribute to the evidence model. In turn, both should be separated from those variations that matter only for presentation purposes.

Deane and Sheehan (2003) and Higgins, Futagi, and Deane (2005) focused on one particular limitation of a template substitution system: that it does not support automatic generation of more varied syntactical structures. This limitation is not an issue with simple, purely mathematical items such as that shown in Figure 1, but it becomes an issue with word problems, where variations in wording often involve complex changes in grammar and phrasing that cannot easily be handled by templates alone.

Deane and Sheehan (2003) argued that item modeling of math word problems can be more effectively carried out using a more sophisticated natural language generation (NLG) architecture, in which the structure of the language and the language for particular word problem domains are factored out in a three-stage process (cf. Reiter, 1995, Cahill et al., 1999) and is the basis of the word problem Model Creator system reported in Deane (2003), Deane and Sheehan, and Higgins et al. (2005). Stage 1 involves content determination (selecting the abstract content to be delivered without reference to details of document structure, language, or sentence structure). Stage 2 involves sentence planning (deciding how the information will be organized to form a document, but without determining the actual wording). Stage 3 involves surface realization, a mapping from the language-independent representation used in Stage 2 to the actual text. The Model Creator system can most effectively be conceptualized as combining an explicit representation of a distance-rate-time word problem task model with the machinery needed to generate item models and ultimately items from the task-model specification.

The key advantage of this three-stage process is that it distinguishes those aspects of word problems that are language- or domain specific, from those that are (relatively) likely to be related to the task model. That is, we can separate out:

1. Properties associated with an item type as an assessment task, such as the choice of mathematical values and the item type
2. General properties of particular word problem types, such as the vocabulary and typical language associated with distance/rate/time word problems
3. General properties of the target language such as word order and grammatical rules

Point 1 includes most of the elements of word problems that are handled well within the Math TCA framework.

Point 2 is typically associated with what in linguistic theory is called *semantic frame* structure (Fillmore, 1968, 1976, 1977a, 1977b, 1982, 1985; Fillmore & Atkins 1994, Fillmore & Baker 2001.). That is, it is associated with schematic information that can be reduced to general formulae with such forms as PERSON DRIVE VEHICLE DISTANCE, and that is analyzed in linguistic theory as a schematic level of concept representation.

Point 3 involves most of the grammatical and lexical properties of the target language, which means that if these aspects of the item model are separated out from the rest, it is possible to build a system that automatically produces parallel items in multiple languages. Higgins et al. (2005) reported on the results of applying such a system to the generation of word problems in English, Japanese, and Spanish, and showed that it is possible to generate parallel items in multiple languages from a common underlying logical representation.

Deane (2003) presented an outline of elements in Figure 2 that can be isolated through this type of word problem analysis. When this type of analysis is pinned down in detail and instantiated in a NLG system, it yields a three-stage analysis: a document specification, a logical representation, and a text representation. In the prototype interface discussed in Deane, the document specification was realized through menu choices in a user interface; however, the menu choices translate to internally represented XML documents, along the lines shown in Figure 3.

I. Mental Model Structure

Ia	The number of events
Ib	The semantic frame associated with each event
Ic	The primary participants in each event
Id	The identity of primary participants across events
Ie	The secondary roles that are relevant to each event
If	The identity of secondary roles across events

II. Task-Relevant Problem Structuring

IIa	The mapping from mathematical variables to frame-semantic roles
IIb	Which variables are explicitly given values in the text
IIc	Which variables are queried, so that their values must be determined to answer the question
IId	Which other variables are left implicit so that their values can only be determined by inference or calculation
IIe	Additional operations required to complete the task
IIf	Modifications or limitations on variables to guarantee correct solutions

III. Document Format and Structure

IIIa	Document type
IIIb	Use of metalanguage summarizing the relationship among events
IIc	Arrangement of content, e.g., the choice and placement of sentences & phrases directly expressing each event

IV Variations in Language

IVa	Alternative sentence structures
IVb	Alternative methods of identifying referents
IVc	Other variations in detailed phrasing

***Figure 2.* Dimensions of analysis for math word problems.**

This format specifies key information about the document: that is, all the vocabulary is drawn from the distance-rate-time domain; only a single vocabulary set reflecting a single vehicle type is involved; there are two events involving two different vehicles identified by the variables *W* and *X*; the distance information is supplied; and the rate information is asked for, in a quantitative comparison item format. The language is reduced to a specification of what frames (essentially, natural language vocabulary sets) and propositions are involved in the item, and a listing of which specific concepts drawn from these frames are actually expressed in each proposition.

```
          <?xml>
          <body>
                    <variables>
                              W, X, Y, Z, VONE
                    </variables>
                    <frame id="1" frameID="ONE"  type="DRT" VONE="VEHICLE" />
                    <event id="2" frameID="ONE" eventID="A" type="DRT" subj="W@"  rate="30"
          rateVar="Y" />
                    <event id="3" frameID="ONE" eventID="B" type="DRT" subj="X@" rate="70"
          rateVar="Z" />
                    <bindings id="4" frameID="ONE" type="DRT" W="VONE" X="VONE" />
                    <proposition id="2" frameID="ONE" eventID="A" role="QCColumnA" type="DRT"
                              distance="QUERY"
                              rate="EXPRESSED"
                              time="UNKNOWN"
                    />
                    <proposition id="F" frameID="ONE" eventID="B" role="QCColumnB" type="DRT"
                              distance="QUERY"
                              rate="EXPRESSED"
                              time="UNKNOWN"
                    />
          </body>
```

Figure 3. **XML representation of a document specification.**

The NLG system takes document specifications in this format and maps them onto a modified logical representation that captures the base semantic information in schematic form. For instance, the XML specification in Figure 3 translates into a logical expression that contains expressions like the following:

DISTANCE::PERSON|W DRIVEONE(VONE|Y::VEHICLE|Y,RATE=30|Z)

Each of the capitalized expressions (DISTANCE, PERSON, DRIVEONE, VONE, VEHICLE, RATE) corresponds to a word or phrase that will be generated in the final phase of NLG, yielding expressions like:

The distance Jane drove in her automobile at 30 miles per hour.

The key property of this type of NLG system is that it reflects an analysis of the ways in which a specific set of closely related items vary—essentially, a task-model analysis, though some variation is better abstracted and treated as only relevant to the presentation model. The NLG analysis seeks to separate most of the presentation detail from the detail that matters for the task model, by progressively abstracting the content. At the logical level, the details of phrasing have been eliminated, leaving schematic descriptions of sentence content. At the more abstract sentence-planning level, even more detail is left out, with only high-level choices, such as "express distance and rate, query time" being specified. Certain aspects of the XML document specification are still logically part of the presentation model—such as the specification whether the driver or the vehicle is the syntactic subject—but this information is clearly localized in specific XML attributes, leaving the XML document as a fairly transparent representation of those document properties that matter in the task model.

However, the prototype user interface that was used as the front-end to this XML system in Deane (2003) and Deane and Sheehan (2003) has one serious drawback: It directly incorporates both presentation model and task-model analyses in a single interface, exposing them only through a set of menu choices without an explicit connection being drawn to the structure of the task model or to a psychometric analysis. Improving on the word-problem interface requires more flexibility, particularly the ability to separate presentation model from task model far more explicitly and completely, while supporting much more explicit representations of task-model variables. Essentially, what is needed is a general framework for analyzing variation among models, with the kind of linguistic variation in the word problem analysis functioning as one aspect of a much larger analytic system.

Going Beyond Word Problems: Model Analysis of Linear and Simple Rational Equation Items

Structural analysis. The model analysis strategy outlined above can most efficiently be applied when there is a large collection of preexisting items that cover most of the dimensions of variation relevant to a particular class of constructs. Linear and simple rational equations constitute one such construct class, as they are central to high school algebra and play a significant role in assessments that help in making high-stakes decisions. As a key content area with a rich array of items available for analysis, they constitute a natural test case. The critical issue is not merely whether a task-model analysis can be created, but whether the dimensions of variation among items are susceptible to automation. Task-model dimensions may vary. Some may be unique to a specific item or specific to a narrow class. Others may form common dimensions of variation that recur in many different types of mathematical content. The goal of automating the link between task modeling and item modeling will be feasible to the extent that well-defined dimensions of broad applicability can be identified.

For the task-model analysis to be discussed below, 62 multiple-choice linear and simple rational equation items were drawn from the disclosed items available in the *GRE Big Book*

(ETS, 1996). Here the term *simple rational equation* refers to a rational equation that may be expressed as a linear equation (e.g., variables are of first degree; denominators are non-zero). We limited our analysis to pure math items (word problems that described a situation were not included).

These were then analyzed to determine the major dimensions of variation of the sort that would be included in a task model. The analysis worked backward from existing items to infer task-model data, without performing a full ECD analysis back to inferred constructs and skills. The goal was to identify the task-model relevant structure of the items as a necessary preliminary step, with a focus on identifying ways in which variation at the task-model level could be controlled and automated.

Four major dimensions were isolated:

1. Problem goal
   a. Solve for a single variable
   b. Find a function of the variable(s)

2. Number of equations
   a. One equation
   b. Two equations

   One equation means that one equation is provided in the stem of the item. *Two equations* means that two equations are provided in the stem. For both symbolic and verbal stem representations, simple assignment statements of a value to a variable (e.g., $x = 5$) were not counted. Anything more abstract, however, like function(variable) = function(variable), function(variable) = value, variable = constant term, or variable = variable, was counted as an equation.

3. Key format
   a. Numeric
   b. Symbolic

   Key format refers to whether or not the key is expressed in terms of a variable. Enright and Sheehan (2002) explored the impact of key format on item difficulty of GRE algebra and arithmetic items.

4. Stem format
    a. Symbolic
    b. Verbal

Figure 4 shows some examples of items classified by these dimensions.

1. Solve for a single variable, one equation, numeric key, symbolic stem:[4]

If $\frac{x}{2}+1=15$ then x =

(a)	5
(b)	7
(c)	13
(d)	28
(e)	29

2. Find a function, two equations, symbolic key, symbolic stem:[5]

If $y=3x$ and $z=2y$, then in terms of x, $y+z=$

(a)	10x
(b)	9x
(c)	8x
(d)	6x
(e)	5x

3. Solve for a variable, two equations, numeric key, verbal stem:[6]

If one number exceeds another number by 13 and the larger number is $\frac{3}{2}$ times the smaller number, then the smaller number is:

(a)	13
(b)	26
(c)	31
(d)	39
(e)	65

***Figure 4.* Three example items and their structural classifications.**

The items were distributed as shown in Table 1.[7]

Table 1

Classification of Linear and Simple Rational Equations

Type	# of equations	Problem goal	Key format	Stem format	# of items
1A	1	Solve for variable	Numeric	Symbolic	15
1B	1	Solve for variable	Numeric	Verbal	1
2A	1	Solve for variable	Symbolic	Symbolic	8
2B	1	Solve for variable	Symbolic	Verbal	1
3A	1	Find function	Numeric	Symbolic	12
3B	1	Find function	Numeric	Verbal	5
4A	1	Find function	Symbolic	Symbolic	0
4B	1	Find function	Symbolic	Verbal	1
5A	2	Solve for variable	Numeric	Symbolic	5
5B	2	Solve for variable	Numeric	Verbal	3
6A	2	Solve for variable	Symbolic	Symbolic	2
6B	2	Solve for variable	Symbolic	Verbal	0
7A	2	Find function	Numeric	Symbolic	3
7B	2	Find function	Numeric	Verbal	1
8A	2	Find function	Symbolic	Symbolic	1
8B	2	Find function	Symbolic	Verbal	0

Note. Data based on an analysis of items from the *GRE Big Book* by ETS, 1996, Princeton, NJ: Author. Copyright 2005 by ETS. Used with permission.

This analysis indicates a clear preference for one-equation over two-equation items (43 out of 58 items), for solving for a variable over find a function (35 out of 58 items), for numeric over symbolic response format (45 out of 58 items), and for symbolic over verbal representation (46 out of 58 items). Type 1A thus represents the modal equation item type.

Impact of Structural Attributes on Item Difficulty

Exploratory regression analysis. This classification scheme was primarily designed to capture structural differences among the set of 58 items. Nevertheless, some of these structural attributes may impact item difficulty. To determine which attributes might also be important predictors of item difficulty, we performed a stepwise multiple linear regression. Due to the small number of items for some levels of attributes and the fact that these attributes were not designed as difficulty predictors but were included as part of a structural classification scheme, this analysis should be considered strictly exploratory.

Four predictors were included as candidates in the regression, as follows: number of equations (0 = one equation, 1 = two equations), key format (0 = symbolic, 1 = numeric), stem format (0 = verbal, 1 = symbolic), and problem goal (0 = find function, 1 = solve for variable). P+ was regressed on the levels of the predictors entered at each step. In order for a predictor to be entered, the p-value of its associated F-statistic had to be less than .05; in order for the predictor to be removed on a subsequent step, the p-value of its F-statistic had to exceed .10. The results from each step of the regression are shown in Table 2; the last two columns show the zero-order and partial correlations between each predictor and the dependent variable.

Table 2

Stepwise Regression Analysis Predicting P+ From Key Format, Number of Equations, and Stem Format

Predictor	B	SE B	β	R^2 change	Adjusted R^2	Correlations	
						Zero-order	Partial
Step 1							
Constant	58.92	4.08					
Key format	19.17	4.64	.48**	.23**	.22	.48	.48
Step 2							
Constant	61.33	4.03					
Key format	19.54	4.44	.49**			.48	.51
# of equations	-10.43	4.23	-.28*	.08*	.29	-.26	-.32
Step 3							
Constant	53.36	5.50					
Key format	20.14	4.33	.51**			.48	.54
# of equations	-9.71	4.12	-.26*			-.26	-.30
Stem format	9.22	4.47	.23*	.05*	.32	.21	.27

**p* < .05. ***p* < .001.

Three of the predictors were entered into the regression, in the following order: (a) key format, (b) number of equations, and (c) stem format. None of the predictors were removed on a subsequent step. The problem goal predictor was not entered into the regression because it did not meet the significance criterion (*t* = 1.43, *p* = .16). This suggests that the problem goal feature did not have a significant impact on difficulty. The mean P+ for items that involved solving for a variable was 74.26 (*n* = 35, *SD* = 15.36), and the mean P+ for items that involved finding a function was 73.09 (*n* = 23, *SD* = 18.83).

Regression results and cell means. For the predictors that were entered into the regression, it appears that items with symbolic keys are more difficult than items with numeric keys, items with two equations tend to be more difficult than items with only one equation, and items with verbal stems may be more difficult than items with symbolic stems. Table 3 shows the mean P+ for each combination of levels from the predictors that were entered in the final step of the regression (each number of equations by key format by stem format combination is shown). There is one empty cell in the table, because there were no items with two equations, symbolic key format, and verbal stem format in the item set. For each predictor, it is still possible to make three pairwise comparisons between means. An examination of the means in Table 4 echoes the results of the regression analysis. It should be noted, however, that there are relatively small numbers of items that have:

1. Symbolic keys (13 items)
2. Two equations (15 items)
3. Verbal stems (12 items)

Key format. The finding that items with symbolic keys are more difficult is not surprising, given results from earlier work. In a study on rate word problems, Enright et al. (2002) found that introducing a variable contributes significantly to item difficulty. The rate problems explored by Enright et al. had either a single variable or only numeric values, both of which appeared in the stem and the key. The items analyzed in the current study differ in a number of respects: First, they are not word problems; second, they vary more widely in structure; and third, the use of a variable in the stem does not imply the use of a variable in the key. In spite of these differences however, the symbolic key items in the study by Enright et al. and the symbolic key items in the current study require that a correct response be expressed in terms of a variable, and this alone is likely contributing to item difficulty. Another study by Enright and Sheehan (2002) explored factors that influence difficulty, for a broader spectrum of GRE algebra and arithmetic items. They found that for items with certain attributes, the form of the item solution impacted difficulty (i.e., items that had quantities as solutions were easier than items that had mathematical or logical expressions as solutions).

Table 3

P+ Values by Structural Attribute Level

	Stem format											
	One equation						Two equations					
	Symbolic			Verbal			Symbolic			Verbal		
Key format	Mean	Min-max	*n*	Mean	Min-max	*n*	Mean	Min-max	*n*	Mean	Min-max	*n*
Numeric	82.52	63–95	27	77.17	53–93	6	70.38	31–94	8	65.00	52–86	4
Symbolic	63.88	44–74	8	40.00	28–52	2	58.33	33–71	3	.	.	0

Note. Mean, minimum, and maximum P+ values for combinations of levels are given. The *n* column indicates the number of items upon which each statistic is based. There is one empty cell because there were no items with two equations, symbolic key format, and verbal stem format.

Number of equations. The finding that two-equation items are more difficult than one-equation items was also anticipated. Sebrechts et al. (1996) studied a sample of 20 algebra word problems from the GRE quantitative exam. One strong predictor of item difficulty was the *nest level*, which referred to “the numbers of levels of parentheses or the number of equations in the representation” (p. 300). Sebrechts et al. found that higher levels of nesting were associated with lower levels of performance. In the current study, nest level has a somewhat different interpretation for items with symbolic stems, because the representation (in this case an equation or system of equations) is given in the stem, and the requirement is that the examinee operates on the provided representation. For items with verbal stems, the meaning is closer, but nest level was not among our high-level structural classifications. Still, the number of equations variable addresses one form of nesting; giving an additional equation in the stem might reasonably be expected to make an item more difficult.

Stem type. It is not clear why items with verbal stems were somewhat more difficult than items with symbolic stems. There have been controlled experiments that have investigated the impact of stem format on student performance on linear equations items. For example, Mayer (1982a) explored the impact of stem format on response time for solving linear equations in one variable. College students who saw problems in equation format (what we refer to as *symbolic stem* format) solved them more quickly than college students who saw equivalent problems in word format (what we refer to as *verbal stem* format in our classification). Because the experiment focused on response time, participants were selected to be able to solve the equations (participants who did not solve a trial equation were replaced).

Koedinger and Nathan (2004) found that algebra high school students can perform better on problems presented in word equation format than on mathematically equivalent problems presented in symbolic equation format, at least when the language used is natural and familiar. Both the Mayer (1982a) and the Koedinger and Nathan experiments involved samples drawn from different populations than the one explored in this study, which was based on the performance of GRE examinees. In addition, the current analysis is not based on a controlled study, so it is possible that other item features that are correlated with stem format are actually responsible for the apparent impact on difficulty. Finally, the items explored in this study are different with respect to content and format (the items investigated here spanned a broad range of item structures, and all were in 5-choice multiple-choice format). In sum, it is not clear how to interpret the impact of stem format on difficulty in the current analysis.

An additional wrinkle concerns the specific wording of items with verbal stems. As noted by Koedinger and Nathan (2004), differences in how the language is expressed can impact problem difficulty. Lewis and Mayer (1987) found that word problems that use *inconsistent language* are more difficult to solve than word problems that use *consistent language*. For example, if a student is told that Gwen has six candies and that she has two more candies than Stella, the student subtracts to find out how many candies Stella has in her possession. This example illustrates the use of inconsistent language because the phrase *more than* suggests an addition operation rather than a subtraction operation.

Although it is an algebraic translation item rather than a numeric word problem, the famous "students and professors" statement (Clement, Lochhead, & Monk, 1981, p. 288;

Clement, Lochhead, & Soloway, 1979, Table 1) also uses inconsistent language. The statement is as follows:

> Write an equation using the variables S and P to represent the following statement: "There are six times as many students as professors at this University." Use S for the number of students and P for the number of professors.

Among college students, a common incorrect response to this item is $6S = P$ (this is known as the *variable reversal error*). It is important to note here that the terms *consistent* and *inconsistent* do not refer to the internal consistency of the problem statement; they refer to whether or not the correct algebraic representation is consistent with a direct word order translation. Algebraic representation problems that are direct translation-inconsistent are more difficult than corresponding numeric word problems. For example, Martin and Bassok (2005) found that college students have no difficulty solving word problems such as the following: "There are 3,450 students. If there are six times as many students as professors, how many professors are there?" (p. 474, Table 1). Presenting students with a word problem to solve prior to translating a statement to an algebraic equation improves performance (Borchert, 2000), but there is evidence to suggest that students do not clearly understand the relationship between solving word problems and representing algebraic equations (Borchert, 2003). Also, algebraic representation problems that are direct translation-consistent are much easier than algebraic representation problems that are direct translation-inconsistent (Graf, Bassok, Hunt, & Minstrell, 2004).

Given these complexities—the small number of items with verbal stem format, and the fact that this analysis is not based on results from a controlled experiment—it is open to speculation why verbal stem items were more difficult than symbolic stem items. One possibility is that the finding is specific to the particular item set under study. Another possibility is that GRE population examinees are relatively experienced with symbolic representations and find them easier to work with than verbal representations. Yet another possibility is that items with verbal stems may be more difficult because some of them are direct translation-inconsistent. We inspected the 12 verbal stem items more closely and classified each of them according to whether they were direct translation-inconsistent or direct translation-consistent. Two of the items were not classifiable in this respect, five were direct translation-inconsistent, and five were direct translation-consistent. The mean P+ for the five direct translation-inconsistent items was

64.60 (*SD* = 16.65), and the mean P+ for the five direct translation-consistent items was 80.00 (*SD* = 12.43). The number of items is small and the difference between these means is not statistically significant, but it is in the expected direction.

Problem goal. For this set of items, no effect of problem goal was apparent. One might expect that find function items would be more difficult if examinees tended to solve explicitly for the variable as a preliminary step. For a number of the items in which a function is required, however, it is not necessary to solve explicitly for the variable(s) in the stem. For example, item #2 in Figure 4 requires composition rather than solution, and in this case the composition is straightforward. In general, it is not uncommon that the required function is a linear combination of the equation(s) presented in the stem, and to the extent that this is recognized, these items test composition skill rather than equation-solving skill. Items that may be solved by extremely simple composition may actually be easier than items that require explicit solution.

Summary and interpretation. The results for the impact of key format and number of equations on difficulty are consistent with findings described in prior literature. Items with verbal stems were somewhat more difficult than items with symbolic stems. The reasons for this are not clear. It is interesting that the mean P+ for direct translation-consistent items was higher than the mean P+ for direct translation-inconsistent items, although this difference was not significant. Further experimental studies should explore differences in difficulty among items that use symbolic stems, verbal stems with consistent language, and verbal stems with inconsistent language, for examinees at different levels of equation-solving expertise. In follow-up studies, items should be controlled with respect to other attributes. There was no evidence that problem goal had an impact on difficulty, but since this was not an experimentally controlled study, it is possible that other characteristics of these items mask its influence.

The results of the regression analysis show that some of the structural variables in our classification scheme appear to have an impact on difficulty. But it is equally important to note that most of the variance in difficulty is still unaccounted for—the three predictors together still only account for 32% of the variance.[8] To an extent, this may be due to the fact that our structural classification is broad rather than fine-grained. For example, within each of these structural classes, some of the items contain parenthetical expressions and some do not. Sebrechts et al. (1996) considered this as part of the *nest level* attribute, but we did not examine its impact here. Payne and Squibb (1990) analyzed the incidence of algebra mal-rules (i.e., procedural errors; see Sleeman, 1984) among students solving linear-equation items. The items they examined are similar to the items contained in our Type 1A category—the stem format was symbolic, there was one equation and one variable, and the task was to find the value of the variable, which was numeric. Although they found that mal-rules were idiosyncratically applied, a couple of the more common ones that they noted involved operating on parenthetical expressions.

There are undoubtedly other features that influence difficulty that are not structural at all. For example, whether quantities in the stem and the key are positive, negative, integers, or fractions may impact item difficulty. Number type has been shown to influence algebra high school students' performances on algebra problems (Koedinger & Nathan, 2004) and elementary school teachers' performances on multiplicative arithmetic problems (Harel, Behr, Post, & Lesh, 1994). Enright and Sheehan (2002) found that for GRE quantitative items that are *procedural* and have no *nonlinear constraints* (i.e., working with nonlinear representations is not necessary), items with more fractions or decimals in the stem are also more difficult. In general, properties of numbers that make intermediate computations easier, or that allow shortcuts, are likely to influence difficulty. Finally, for multiple-choice items, properties of distractors (both structural and numeric) are likely to influence item difficulty. All of these are features that are not accounted for in a purely structural classification, though they are features that will need to be captured in a detailed task model.

Implications. The results of the preceding analysis suggest that for the purpose of characterizing task difficulty, it may be useful to devise a framework that incorporates the potential impact of both structural and nonstructural features. One way to conceptualize the problem is to think of the space of possibilities as an *n*-dimensional grid, with the structural and nonstructural features functioning as dimensions defining a large multidimensional space. To the extent that these features can be correlated with difficulty, individual cells will represent single tasks with predictable difficulty. Of course, only a small subset of the cells in such a large hyper-dimensional grid will actually correspond to desirable tasks with valuable psychometric properties. Part of the problem in specifying appropriate task-model variables for a domain is to identify what those tasks are and to provide computational support for producing items and item models with desirable combinations of features.

It is important to note that we are not concerned merely with enumerating features to be listed in a difficulty model, nor with merely providing an item-generation system with the ability to solve the system of equations mechanically to identify answers or item variants, but with the ability to manipulate systems of equations in terms of features that matter both from a task-model perspective and in terms of the student model.

This requires that we manipulate mathematical content in ways not supported by a template-based presentation system. In Math TCA, the equations that define the basic item content are present in the constraint definition in some form, but the only purpose of the constraint representation is to support calculation of possible values for each variable. Part of the specification of a class of items in a task model is a high-level specification of the types of mathematical content to which the task model is relevant. Software that supports such a specification will make use of the relationships between mathematical constraints and the cognitive properties of item models. This implies a design requirement: The user should be able to specify meta-level constraints that capture the conceptual similarities among related item models.

Similarly, the first and third task-model features—(a) solving for a variable versus a function of variable(s), and (b) numeric versus symbolic response—correspond at an abstract level to one of the basic elements of any task model: specifying what relationship responses to the task should have to presented content. Here again, we may draw an important distinction between the requirements of item modeling in isolation, and those that must be met in an explicit task-modeling framework. The distinction between solving for a variable and solving for a function could be represented in Math TCA simply by providing a single variable, or an expression, as the key. However, such a low-level specification of actual response values does not provide control of the nature of the task at a level of abstraction appropriate to specifying task-model variables. If a system could be devised that supported the generation of item models by specifying task-model properties, the natural relationship would be to specify the type of response and for that decision to constrain the ways in which item-model responses could be authored.

Finally, the fourth major task-model feature represents another fundamental way in which tasks can be varied: the means by which content is presented. In this case, the variation has to do with the choice of verbal versus algebraic presentation of the core math content, but other forms of variation, such as graphical presentation, are possible, and in many cases, could be automated. Further examination of the ways in which linear-equation items of the same type vary among themselves suggests that such forms of variation may include, most significantly:

1. Variations in the form of the equation
2. Variations in the wording of the stem
3. Variations in how the distractors are represented
4. Variations in the numeric values included in the equation
5. Interactions between the numeric values chosen and the form of the equation

Our concern in what follows will not be with a detailed examination of the effect of these kinds of variations on item difficulty and thus with the precise formulation of a student model, but rather with considering the extent to which arguably task-relevant features can usefully be manipulated within an appropriately configured software system. The kinds of variations we are concerned with here are those that clearly could affect the nature of the task, and that could therefore have ramifications not only with respect to the student model and an account of item difficulty but that also could help to determine what variations in presentation and content were appropriate variations of the same underlying task model.

Variations in the form of the equation. The form that linear and simple rational equations can take is not fixed, as they can vary along a number of dimensions. The simplest form that a linear equation can take will place the variable on one side and a constant on the other:

$$x = 5$$

The variable may have a coefficient:

$$3x = 5$$

And there may be a constant on the same side of the equation as the variable:

$$3x + 1 = 5$$

There may be distribution of a constant across multiple terms:

$$3(x + 1) = 5$$

There may be a summation of terms containing the same variable:

$$3x + 2x = 4$$

There may be the use of subtraction instead of addition, for the relationship between terms:

$$3(x - 1) = 4$$

And the equation may be rearranged so that variables appear on the right side as well as the left side of the equation:

$$3x + 2 = 4 - x$$

There can also be information in the denominator on either side (or both):

$$\frac{3x+2}{5} = 4 - x$$

$$3x + 2 = \frac{4-x}{3}$$

And this information can be distributed among the terms:

$$\frac{3x}{5} + \frac{2}{5} = 4 - x$$

Moreover, a variable may appear in the denominator (this is a simple rational equation):

$$\frac{5}{x} = 3$$

And, of course, various combinations of these can be produced by all the standard algebraic operations on equations.

These kinds of variations can be viewed as essentially similar in kind to the distinction between one- and two-equation items—they involve manipulations of the mathematical content, either at the level of specifying what that content is or of casting it in a particular algebraic form.

Note that this kind of mathematical content (as expressed in a system of equations) actually has to play multiple roles. We need to know the system of equations for item-modeling purposes, because we have to be able to determine which sets of alternative values constitute valid item variants. We need to know the system of equations for scoring purposes, because we need to distinguish between keys and non-keys. But we also need to know how to present the system of equations, whether in formulae, equations, charts, diagrams, or other presentation modalities, and we need to be able to cast it into mathematically equivalent forms that may vary significantly in their task requirements. Each of these applications represents requirements as to the use of mathematical content that may impose different requirements on a computational implementation. For instance, the representation of math content in Math TCA appears to be optimized for interaction with its built-in constraint engine, the main function of which is to solve the equation and produce values for all variables that are consistent with a model's mathematical constraints. A different mode of representation—perhaps presentation of MathML—will be needed to support presentation of math content in the stem of an item, and yet other modes may be necessary depending on the use to which mathematical content is being put. Each of these manipulations of mathematical content has potential to impact the student model and the evidentiary value of an item, and so helps create a complex *n*-dimensional grid of possibilities.

Variations in the wording of the stem. Linear/simple rational equation items, of course, are purely mathematical items where the kinds of variation in phrasing that are normal for word problems are strictly ruled out. The language has to be precise enough to specify the assessment task clearly, which means, in turn, that the range of variation is, of necessity, limited to such paraphrase relationships as will preserve the basic character of the item without any change in mathematical content. As a result, the kinds of stem phrasings that can be observed with linear/simple rational equation items are precisely the sort to which the template-based strategy of Math TCA interface is relatively well suited. However, there are patterns in the way linear questions are worded that suggest that choice of wording must be accounted for at the task-model level, and thus support a somewhat more complex model of how the wording of stem content should interact with mathematical content.

The choice of wording in the stem of a linear/simple rational equation item appears to vary partially by the class of item and partially independently. In Type 1A items, for instance, the standard form of the stem is:

If <equation>, then <variable or expression> =

that is, a typical example would be:

$$\textit{If } 3(x+1) = 4x - 1\textit{, then } x =$$

Certain aspects of this form are dictated by the nature of the item; that is, given that Type 1A gives an equation and asks for the value of a variable, obviously both must be mentioned in the stem. However, other stem forms are possible. For instance, there are some problems with the form:

What value of <variable> satisfies the equation <equation>?

And other variations are possible, such as

If <equation>, what is the value of <variable or expression>?

For instance,

If $4x\text{-}2y = 8$ *what is the value of* $2x - y$ *?*

From an ECD perspective, variations as small as this are potentially relevant to the student model. For instance, the question form that asks whether an equation is satisfied presupposes that students understand what it means for a value to satisfy an equation, which means that the difference between different words of the stem may cause a difference in the value of the evidence, depending on whether the knowledge it presupposes (in this case, knowledge of mathematical vocabulary) is construct relevant.

What this means, in turn, is that software implementing model creation must support many-to-many pairings of mathematical content with question wordings, even when the wordings in question are entirely formulaic and seem equivalent in content. Changes in wording are (possibly subtle) changes in task, and so must be accounted for. It would be inappropriate however, to allow completely unconstrained combinations of wording with content, as there are clear dependencies between content and the manner in which questions can be phrased.

For instance, variations from the wording pattern for Type 1A seem to be linked strongly to the type of item. Thus, in two-equation items for which a symbolic response is required, the wording is altered to make it clear that the answer is to be stated in terms of another variable:

If <equations>, what is <variable> in terms of <variable(s)>?

If <equations>, then in terms of <variable(s)>, <expression> =

In cases where numeric information is provided in algebraic form, but assigned a unique value, the form can be extended using expressions such as *when*:

If <equations>, what is <variable> in terms of <variables> when <expression>?

The greatest variations appear in the items where the equations are expressed verbally. Most of them follow one of the standard forms with simple alterations such as using the word *is* instead of the equal sign, but there are more complex forms, none of which appears more than once in the sample. We find, for instance, such patterns as:

If <expression>, <value> would be <expression>.
Therefore, <variable> equals

<equation>. In terms of <variable>,
which of the following is <expression>?

<equation>. If one of the numbers is <variable>,
the other number must be

What seems clear is that the choice of wording for the stem is a function primarily of the major task-model variables, with some variation within single categories, and thus we need to support a model that allows (constrained but at least partially free) combinations of wording with content.

Variations in distractor selection strategy. Because the family of linear/simple rational equation items examined in this study is multiple-choice GRE problem-solving items, the choice of distractors plays an important role in item modeling and in item difficulty. An approach such as facet-based assessment can be used to inform how to represent item distractors (Hunt & Minstrell, 1994; Minstrell & Hunt, 2004; Thissen-Roe & Hunt, 2004).

In facet-based assessment, item design is built around a set of conceptions and misconceptions that students may entertain with respect to particular instructional concepts, termed *facets*. In physics, for instance, students may entertain a series of ideas about how to calculate average speed, ranging from a completely correct understanding to a series of misunderstandings that will sometimes yield the correct answer, such as:

average speed = final position / final time

average speed = change in position / total time

average speed = (initial speed + final speed) / 2

average speed = final speed / 2

(Minstrell & Hunt, 2004, pp. 10-11).

Each of these facets represents both a common error pattern and an associated cognitive state. In facet-based assessment, items are constructed that systematically test the facets associated with a concept. That is, distractors are chosen specifically to give students the opportunity to demonstrate which misunderstandings (or correct interpretations) they entertain about each topic of instruction. The tests are used as formative assessments and are used by teachers to determine what instructional interventions are appropriate for particular students or groups of students.

Similar considerations should be applied to item modeling in general. Graf et al. (2005) focused in particular on the problem of creating item models for linear inequalities and discussed a number of instances where the item characteristic curves for multiple instances of an item model fail to display parallel item operating parameters. Instead of clustering tightly around the parent model upon which they were based, there are subsets of models with significantly different levels of difficulty and discrimination. Analysis suggests that the lack of homogeneity is due to the fact that some instances of a model have options that correspond to attractive misconceptions—facets of the underlying instructional model—whereas others do not.

The implication for model analysis is that distractor analysis is a critical element of the entire process, but that the presence of distractors significantly complicates task analysis. Facets—or student misconceptions—may be salient in a particular item because of a variety of factors, some of them hinging upon important task-model variables, and others upon specific features of the item as presented (such as the exact numeric values presented in a specific item). The implication is that the choice of distractors is likely to interact with a variety of factors in an item model or even an item level. It is an open question whether one could create a single pool of distractor strategies that could be used interchangeably within a single cell in the task-model grid, much less across cells. The implication is that multiple-choice items may be less fully explicable in terms of a small set of common task-model variables than other item types, and that a significant effort may be required to map out the kinds of misconceptions that can drive distractor selection.

An exploratory analysis was performed for the Type 1A category of linear/simple rational equations by constructing a set of item models in Math TCA, basing them upon the 15 items of this category in the data set. The actual distractors chosen for each item were used to create tentative hypotheses about the kinds of models that could underlie them. Of course, there is no guarantee that the distractor models built in this fashion fully capture the crucial properties of the original items, but they do indicate what sort of hypotheses can easily be formulated to account for the distractors chosen in the original item. It was possible to model the distractors in terms of three primary categories:

1. Values spanning a range close to the key
2. Coefficients and constants from the equation
3. Values corresponding to incorrect combinations of the coefficients and constants, such as might be generated in the course of applying algebraic manipulations to the equation

These categories are based on our informal observations about how distractors for linear and simple rational equations are sometimes conceptualized, at least among items in the *GRE Big Book* (ETS, 1996), though these kinds of generalizations are likely to be approximate at best precisely because they can be generalized over a wide range of item types.

As Thissen-Roe and Hunt (2004) indicated, the distractor models used in an item model must be based empirically on analysis of the attractiveness of options over a large number of items from the same domain. While there may be much to be gained by conducting such an analysis over an entire category of items representing a single combination of task-model variables rather than doing it incrementally over individual item models, the potential benefit is far less clear here than with some of the other task-model variables considered above. The implication for the task of constructing software that supports integrated task and item modeling is that it may be premature to do more than what is currently done with distractors in Math TCA—that is, fully supporting the generation of distractors so that they can be specified essentially freely, independent of more general task-model variables.

Variations in the numeric values included in the equation. Differences in the types of numbers used in a mathematical task can affect difficulty; for example, integers tend to appear in easier items than fractions and decimals; smaller numbers tend to appear in easier items than larger numbers; reference fractions like 1/2 or 3/4 tend to appear in easier items than more complex fractions, etc. (National Assessment Governing Board, 2001; Enright & Sheehan, 2002). It is not clear what impact this type of variation has on linear/simple rational equations, but the key point is that there may be a need to support the definition of classes of numbers for item-modeling purposes that correspond to the categories that are likely to have cognitive and thus evidentiary significance. At the task-modeling level, it might be very useful to distinguish (for instance) between items that used only simple, benchmark fractions such as 1/3 or 1/2 and items that required much more complex fraction forms, such as mixed numbers or complex formulas containing fractional elements. In the Math TCA interface, conceptually natural groups (such as benchmark fractions or simple fractions of the form 1/*n*) cannot be specified directly. Thus it would be very helpful to support ways to automate production of variables that fall into ranges corresponding to cognitively natural groups.

Interactions between the numeric values chosen and the form of the equation. Certain numeric values have a major impact upon the difficulty of an item. When a variable has the values 0 or 1, for instance, this may make an item much easier to solve by allowing much simpler solution paths. If two variables are equal to each other, there could be a simpler solution or a simpler equation form based upon obvious algebraic reductions (such as canceling out to one or subtracting out to zero). These kinds of constraints often contribute much of the complexity to an item model, as it is important to write the item model in such a way that easier/simpler versions of an item cannot be produced by accident. This is another situation in which the ease of automation is unclear, but where computational support in item model creation would be very helpful. Mathematical properties that make a solution trivial, to the extent they can be algorithmically identified, would ideally be ruled out by default rather than by requiring model authors to rule them out during the process of writing individual item models.

This preliminary analysis of linear/simple rational equations suggests that many of the kinds of task-model variables one would wish to isolate are fairly general (i.e., applicable to many types of tasks and not just to simple linear equations). It thus makes sense to find ways to automate them, that is, to provide computational support for creating item models in which task-model variable values are explicitly set. To the extent that item models can automatically be derived from task-model variables, there is much to be gained by taking an ECD perspective toward item modeling.

A Prototype System for Model Creation

Workflow Considerations

The considerations adduced so far highlight some of the limitations of the Math TCA as a tool for item modeling. Nothing in Math TCA explicitly supports the links to ECD discussed in the preceding sections. Within Math TCA, item families can be modeled, but this requires explicit programming of individual submodels within a family, and if a family of models corresponds to a particular task-model design, there is nothing in the models themselves that makes that connection explicit. Moreover, the analysis of linear/simple rational equations suggests that variations among purely quantitative models are likely to follow regular patterns, involving common possibilities for variation in equation form, in stem wording, in mathematical value types, and in all the other dimensions that will make up a model grid. In template-based approaches to item modeling, many such variations will have to be modeled individually for each item and item family, which implies the possibility for the massive duplication of effort as more and more item models are written. On the other hand, if item modeling is integrated with ECD, the logical workflow for item development would incorporate several new procedures. Figure 5 presents a simplified version of a workflow for item modeling (Graf et al., 2005, p. 11). In Figure 6, the overall process is presented.

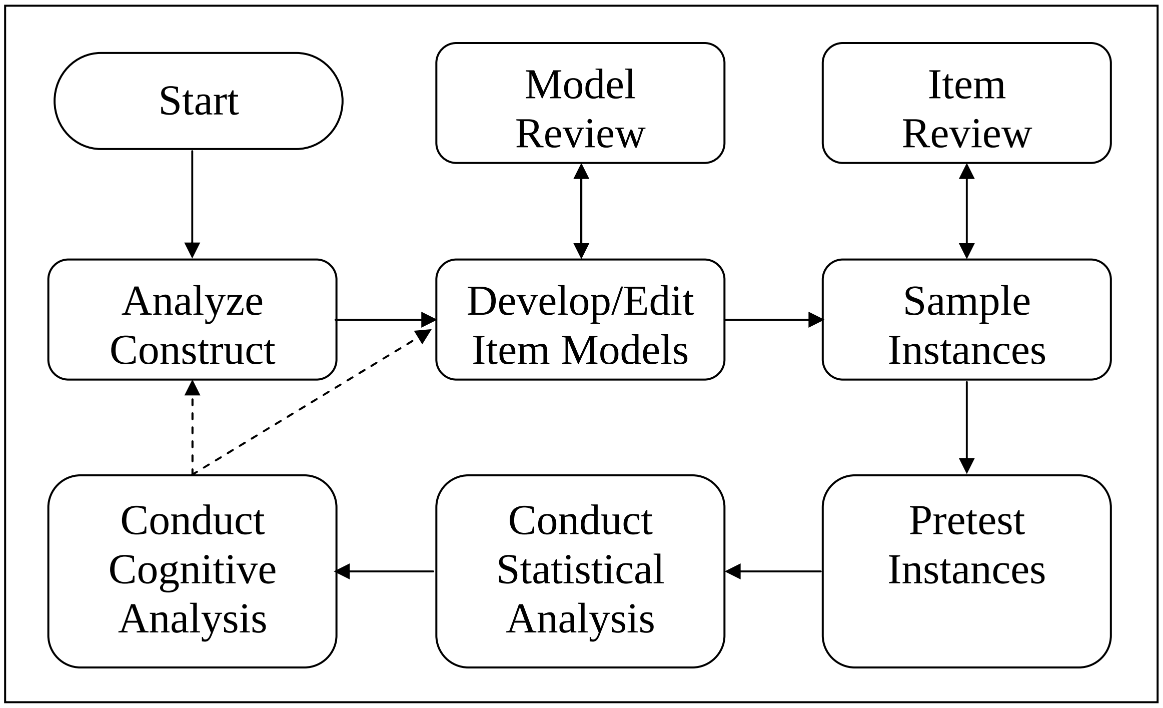


Figure 5. **Item modeling workflow.**

Note. From *Psychometric and Cognitive Analysis as a Basis for the Design and Revision of Quantitative Item Models* (ETS RR-05-25; p. 11) by E. A. Graf, S. Peterson, M. Steffen, & R. R. Lawless (2005), Princeton, NJ: ETS. Copyright 2005 by ETS. Adapted with permission.

Essentially, the *conduct cognitive analysis* and *analyze construct* phases need to be elaborated and related more explicitly to the ECD process discussed above. Figure 6 presents an overview of the resulting workflow.

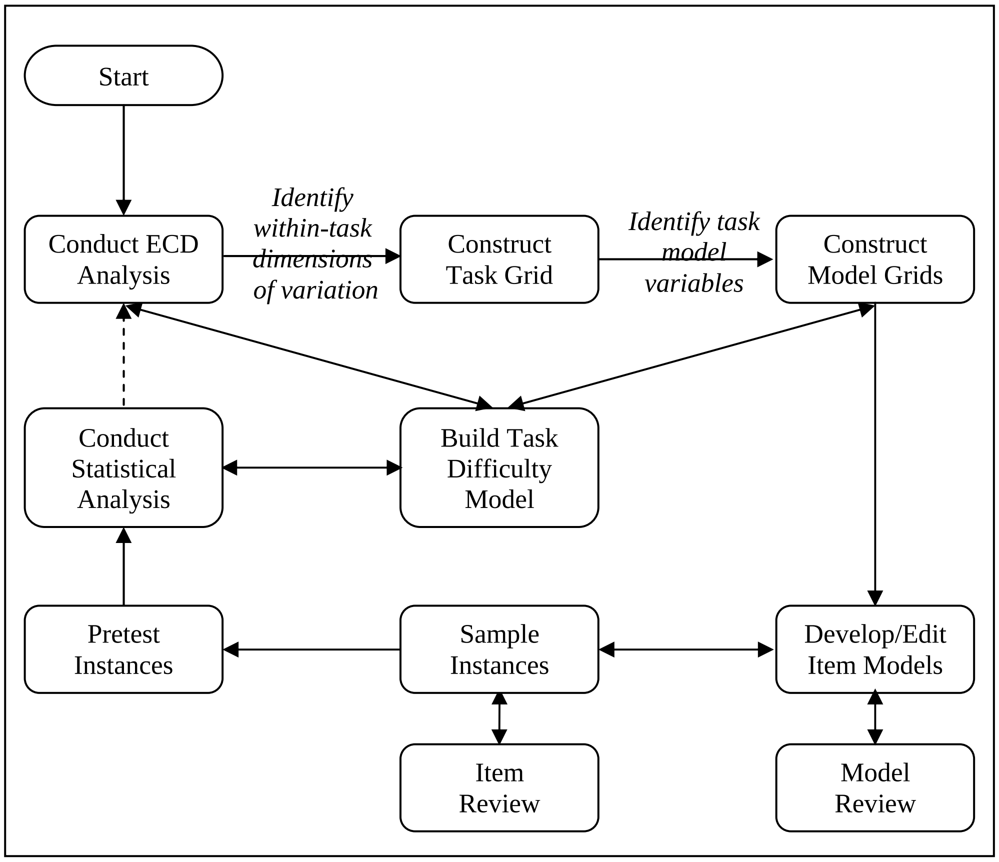


Figure 6. **Item-modeling workflow integrated with ECD.**

Critically, this workflow implies that the task of developing ECD-integrated item models depends upon collaboration among three types of expertise: the expertise test developers bring to the writing and reviewing of items and item models; the expertise that psychometricians bring to the statistical analysis of items and their relationships to variables in a difficulty model; and an expertise in ECD, particularly the ability to analyze a construct to identify the task-model variables and dimensions of variation among items.

Formally, this type of analysis—for example, analysis along the lines presented above—can be conceived of as constructing a multidimensional grid representing different combinations of task-model variables, where each cell in the grid corresponds to a combination of general task-model variables. Let us call this the *task grid*. Each cell in the task grid is then subject to further variation reflecting more idiosyncratic task variations, such as the form of the equation, variations in the distractors reflecting the facets of student knowledge, variations in the computational complexity of the equation, and so on. If these variations are systematically mapped, they will form a second-level grid: what we can term the *model grid*.

Any variations among items that reflect neither a difference in the task grid nor a difference in the model grid are by hypothesis purely presentational variables, and should have minimal impact upon item psychometric attributes, whether they are differences in wording or in numerical values. That is, a single cell in a model grid corresponds by hypothesis to an item model at more or less the level of generality automated by Math TCA (Note that the usefulness of that item model is not guaranteed; not all combinations of task models will have sufficient evidentiary value even to be used, and others may be too difficult, too easy, or otherwise defective.) A task model as described here will only be as useful, from an ECD perspective, as the evidence and student models that it supports. We cannot assume that there will be a simple, transparent mapping from task-model variables to the variables of a formal item-difficulty model. But to the extent that the selection of task-model variables is motivated by ECD considerations, we can anticipate that analysis of entire item pools in task-model terms will provide useful benefits, because it will identify how item models can be constructed in ways that vary consistently with changes in task-model variables.

Perhaps equally importantly, it seems likely that many of the sources of variation at the model-grid level are likely to recur over many different types of items. Variations in the forms of an equation, for instance, and variations in the type of mathematical values plugged into such equations are likely to apply fairly broadly across a wide range of quantitative tasks. For that matter, language-related variations, such as translating mathematical formulae into textual form are also likely to occur in a broad range of situations and could easily be modeled with natural language processing (NLP) techniques. To the extent that variations within a single task can be shown to derive from general processes that apply to many different tasks, it should be possible to automate the task of constructing a model grid, and thus to support ECD-based item modeling across entire content domains much more efficiently.

Another advantage of this approach is that it places earlier work (e.g., Deane, 2003; Deane & Sheehan, 2003) into a larger context. Ideally, word problems and other problems that can usefully be approached using NLG must be integrated with work on item modeling and

ECD. The analysis presented above helps to assess that work and determine how it can best be applied. In particular, it appears that the first stage of the three-stage system outlined in Deane—the document planning stage—has to be included in the task-model analysis; by contrast, the two later stages, which move from document plan to logical representation to text, may be more fruitfully viewed as part of the presentation model. This implies, in turn, that the document design phase needs to be treated as part of the construction of the task and model grids rather than handled separately as part of an NLP system. Because the existing word problem prototype creates an internal representation of document plans as XML documents, it should be possible to separate document planning from the core NLG system, which will allow the NLG to be treated essentially as a presentation module.

To the extent that test developers are able to uniquely identify a particular item model as instantiating a particular design choice in the task and model grids, the items they provide for pretesting can be interpreted by psychometricians as providing evidence for the usefulness of particular variables in the task model, making it possible to develop a detailed task-difficulty model. If the statistical evidence suggests that decisions about the structure of the task or model grid were incorrect, then ECD designers can restructure the ECD analysis to fit the statistical evidence and support test developers' development of new item models that respect the known statistical results.

The key in both directions is encapsulation—that item models should not be open to idiosyncratic modification but should directly reflect design choices during the ECD process, and conversely, that the same variables (and not any smaller, idiosyncratic variations) should be directly available for statistical analysis. The problem, then, is to support a workflow in which experienced test developers are able to apply ECD concepts to build task and item models.

Note that there are existing efforts to support building of ECD systems, such as the Portal project at ETS (Steinberg, Mislevy, & Almond, 2005). The difference between these efforts and the software to be described below is that we seek to provide a way for content developers to explicitly map out the combinations of task-model variables, and simultaneously to support the rapid authoring of item models that instantiate specific tasks. That is, we seek to integrate the construction of item models with the specification of the task model. Ideally in such a model, test developers would be able to make decisions about model structure in terms of design choices rather than all the fine-grained information present in a typical Math TCA-based item model.

A Design for an ECD-Oriented Item Modeling Tool

One way to view the workflow issues is to think in terms of *the resources needed to produce item models for a particular task*. That is, we can think of the dimensions of variation covered in the model grid for a task as providing resources that test developers can draw on, to produce a specific item model. For Type 1A items, this might include the basic form of the equation, a specific wording for the stem, or a particular choice of distractor strategies. What test developers should be able to do is to set up a model at this level of abstraction by choosing which resources their item model will use without having to program every detail. This will only be possible if two things have happened first:

- Test developers have actually created task and model grids.
- The resources that define a model grid have actually been gathered together into resources and loaded into a software interface that allows test developers to create item models without having to program every detail themselves.

These requirements entail further design constraints. Software that supports this kind of model creation will have to be flexible enough to allow most of the ECD specification to be loaded as data, to present that data abstractly enough to allow test developers to think entirely in terms of making design choices, and yet to support automatic creation of item models.

In the remainder of this report, the design for such a tool will be outlined, with detailed specifications of an existing working prototype for quantitative items. The basic concept can be outlined as follows:

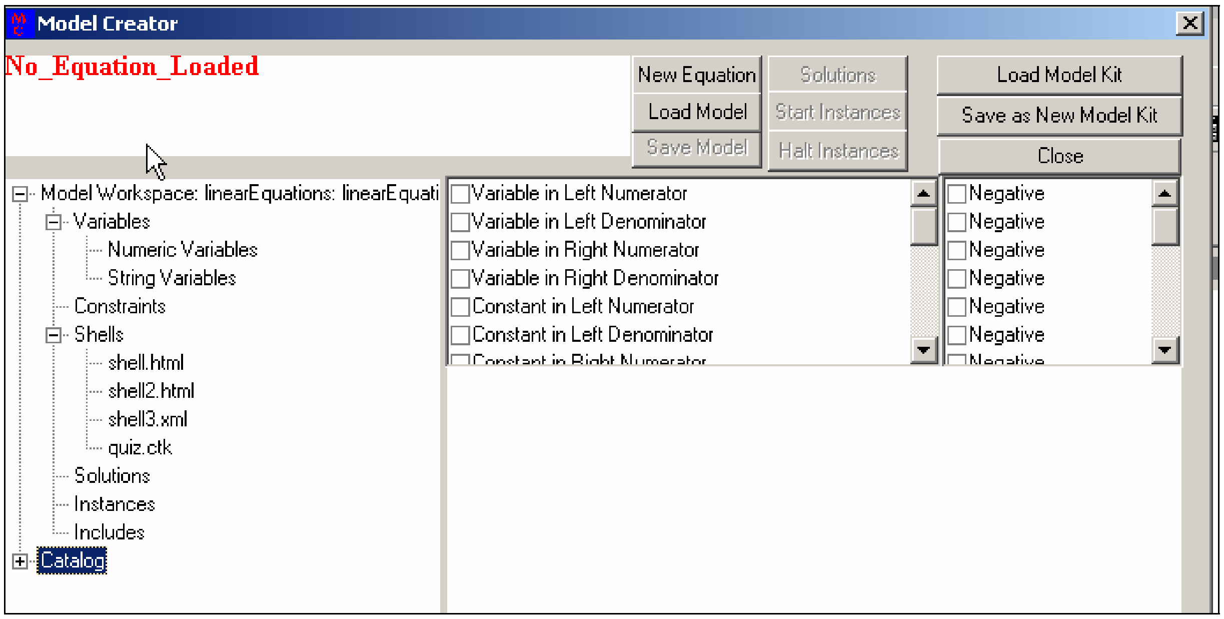
- An XML file format is defined for a file type termed a *model kit*. The model kit defines a set of resources, including most importantly:
    - a *master equation* in MathML format that defines the type of mathematical information that can be included in an item
    - menu choices defining what variations in the form of the master equation are allowed
    - *item shell files* defining the wording and formats allowed for questions
- XML file formats are also defined for *models* and for *model catalogs*. The latter is a file that specifies the locations and relationships among a set of models.

- Software routines are defined to automate a number of processes needed to assemble item models from the resources provided in a model kit, including
    - equation simplification routines that use the master equation and menu choices to produce *equation forms* representing particular variants of the underlying mathematical construct, such as linear/simple rational equations
    - constraint formation routines that convert each equation variant into a set of variables and constraints using the same constraint engine as in the current Math TCA
    - constraint solution routines that call the constraint engine to obtain a set of solutions
    - interpretive routines that allow shell files to be interpreted dynamically by doing such things as plugging variable values into variable positions or evaluating inline functions supplied by the constraint engine
- A user interface is defined that allows users to select the equation forms, shells, constraints, and other variables that define the model grid, and then assemble them to produce instances (model variants) by combining the resources into output files. The interface also supports management of the model catalog and detailed manual modifications of item models.
- The entire prototype is designed so as to be able to support a modular approach to item modeling, in which model kits are prepared in advance for particular tasks (which in the current prototype implementation requires manual creation and modification of XML model kit files), and where test developers create item models by choosing the resources they will deploy in a particular case.

A Prototype User Interface for Model Creation

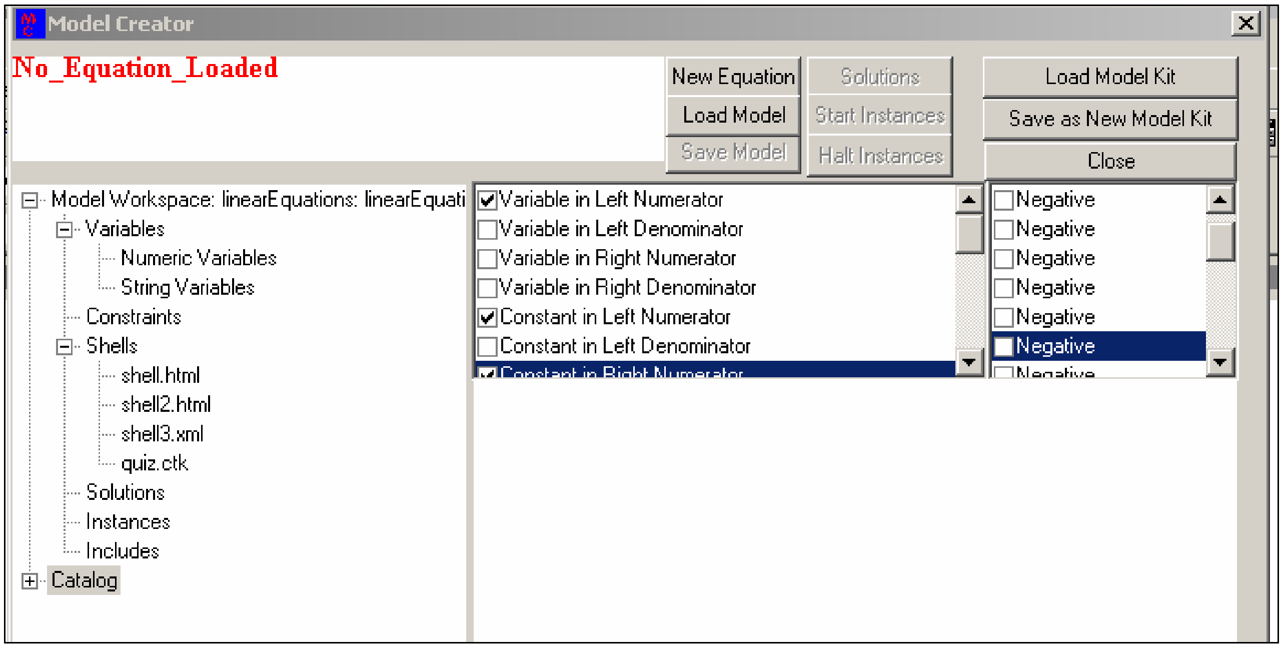
The screen in Figure 7 shows what the interface for model creation looks like after a draft model kit for linear equations has been loaded.

Initially, there is very little information, except a list of shell files (possible presentation formats) and two menus allowing selection of features specifying equation form, including a list of features to include or drop, such as *variable in the left denominator* and a specification of whether that element should take a negative form. Suppose we click the following choices, *variable in the left numerator*, *constant in the left numerator*, and *constant in the right numerator* (as seen in Figure 8).



☐ Variable in Left Numerator — ☐ Negative
☐ Variable in Left Denominator — ☐ Negative
☐ Variable in Right Numerator — ☐ Negative
☐ Variable in Right Denominator — ☐ Negative
☐ Constant in Left Numerator — ☐ Negative
☐ Constant in Left Denominator — ☐ Negative
☐ Constant in Right Numerator — ☐ Negative

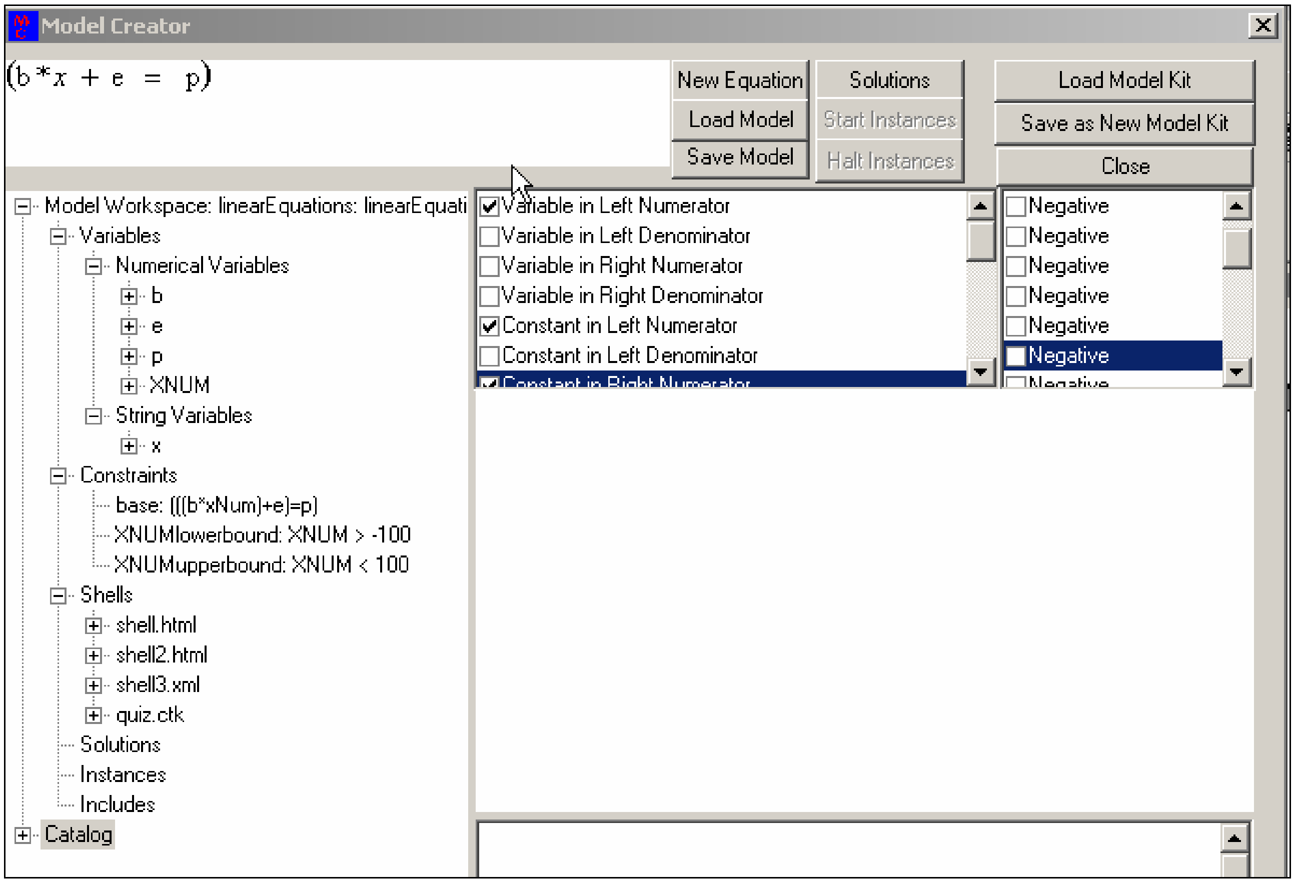
***Figure 7.* Model Creator initial screen.**



✓ Variable in Left Numerator — ☐ Negative
☐ Variable in Left Denominator — ☐ Negative
☐ Variable in Right Numerator — ☐ Negative
☐ Variable in Right Denominator — ☐ Negative
✓ Constant in Left Numerator — ☐ Negative
☐ Constant in Left Denominator — ☐ Negative
✓ Constant in Right Numerator — ☐ Negative

***Figure 8.* Model Creator after selecting equation features.**

At this point, clicking *New Equation* will generate an equation form and a set of variables and constraints to go with it (as shown in Figure 9).



$(b*x + e = p)$

```
Model Workspace: linearEquations: linearEquati
  Variables
    Numerical Variables
      b
      e
      p
      XNUM
    String Variables
      x
  Constraints
    base: (((b*xNum)+e)=p)
    XNUMlowerbound: XNUM > -100
    XNUMupperbound: XNUM < 100
  Shells
    shell.html
    shell2.html
    shell3.xml
    quiz.ctk
  Solutions
  Instances
  Includes
Catalog
```

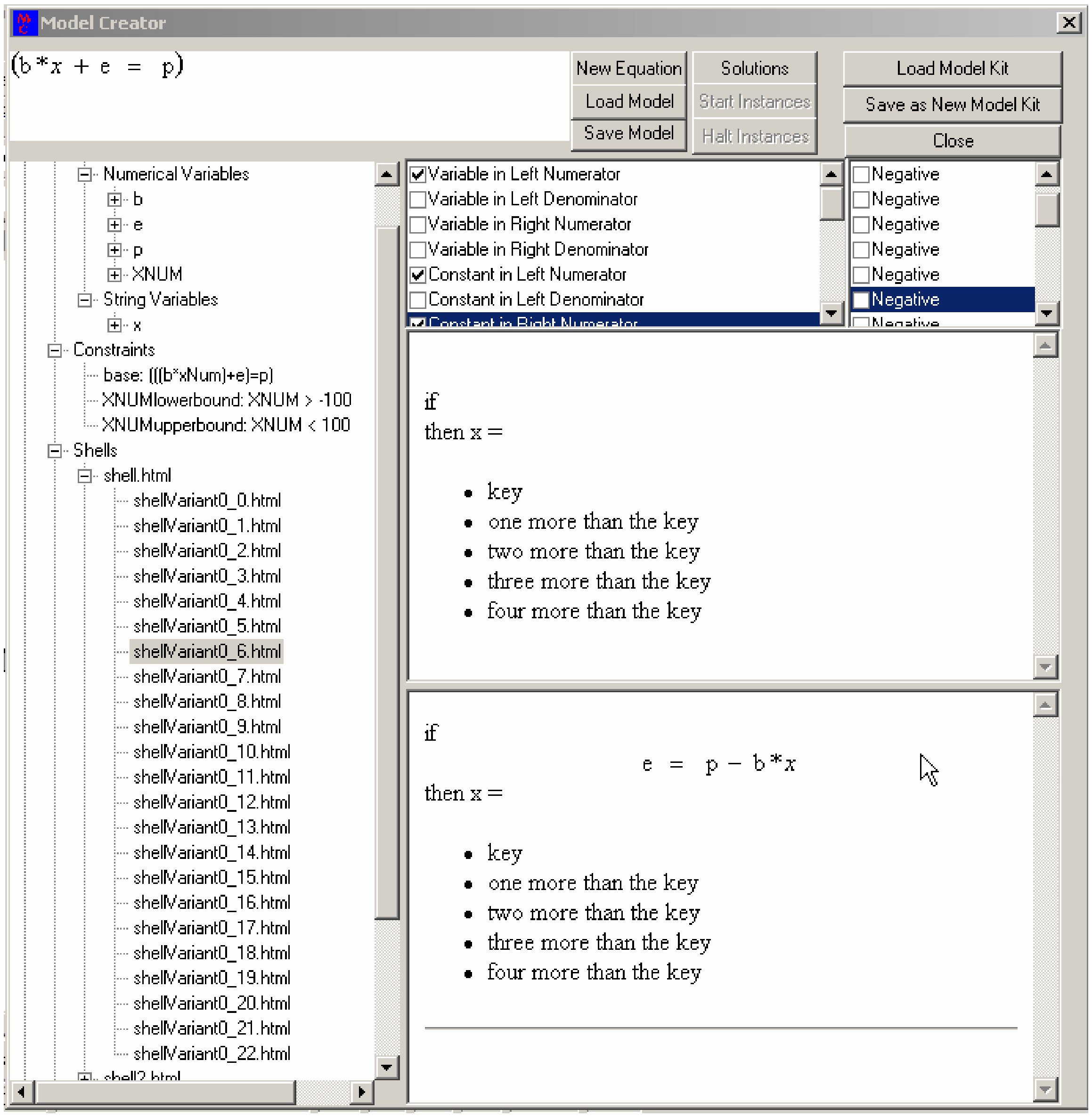
- [✓] Variable in Left Numerator
- [ ] Variable in Left Denominator
- [ ] Variable in Right Numerator
- [ ] Variable in Right Denominator
- [✓] Constant in Left Numerator
- [ ] Constant in Left Denominator
- [✓] Constant in Right Numerator

- [ ] Negative
- [ ] Negative
- [ ] Negative
- [ ] Negative
- [ ] Negative
- [ ] Negative
- [ ] Negative

Figure 9. **Model Creator after generation of equation form.**

At the same time, the system calculates the possible modifications of the base equation form and produces *shell variant* files in which alternate forms of the base equation are combined with shell files. See the example in Figure 10.

In this example, the shell file is shown in the middle pane on the right side of the screen, and an extensive list of shell variants displaying alternate forms of the base equation is shown directly below it, in the bottom pane. In the example shown, the choice of options is specified as part of the shell file and represents a simple default case (providing a set of values near the distractor rather than exploiting specific facets or misunderstandings). Modification of the options is possible but requires that specific choices be made to set up appropriate models for each possible distractor.



$(b*x + e = p)$

```
⊟ Numerical Variables
    ⊞ b
    ⊞ e
    ⊞ p
    ⊞ XNUM
⊟ String Variables
    ⊞ x
⊟ Constraints
    base: (((b*xNum)+e)=p)
    XNUMlowerbound: XNUM > -100
    XNUMupperbound: XNUM < 100
⊟ Shells
  ⊟ shell.html
      shellVariant0_0.html
      shellVariant0_1.html
      shellVariant0_2.html
      shellVariant0_3.html
      shellVariant0_4.html
      shellVariant0_5.html
      shellVariant0_6.html
      shellVariant0_7.html
      shellVariant0_8.html
      shellVariant0_9.html
      shellVariant0_10.html
      shellVariant0_11.html
      shellVariant0_12.html
      shellVariant0_13.html
      shellVariant0_14.html
      shellVariant0_15.html
      shellVariant0_16.html
      shellVariant0_17.html
      shellVariant0_18.html
      shellVariant0_19.html
      shellVariant0_20.html
      shellVariant0_21.html
      shellVariant0_22.html
  ⊞ shell2.html
```

```
[✓] Variable in Left Numerator          [ ] Negative
[ ] Variable in Left Denominator        [ ] Negative
[ ] Variable in Right Numerator         [ ] Negative
[ ] Variable in Right Denominator       [ ] Negative
[✓] Constant in Left Numerator          [ ] Negative
[ ] Constant in Left Denominator        [ ] Negative
[✓] Constant in Right Numerator         [ ] Negative
```

if
then x =

- key
- one more than the key
- two more than the key
- three more than the key
- four more than the key

if

$e = p - b*x$

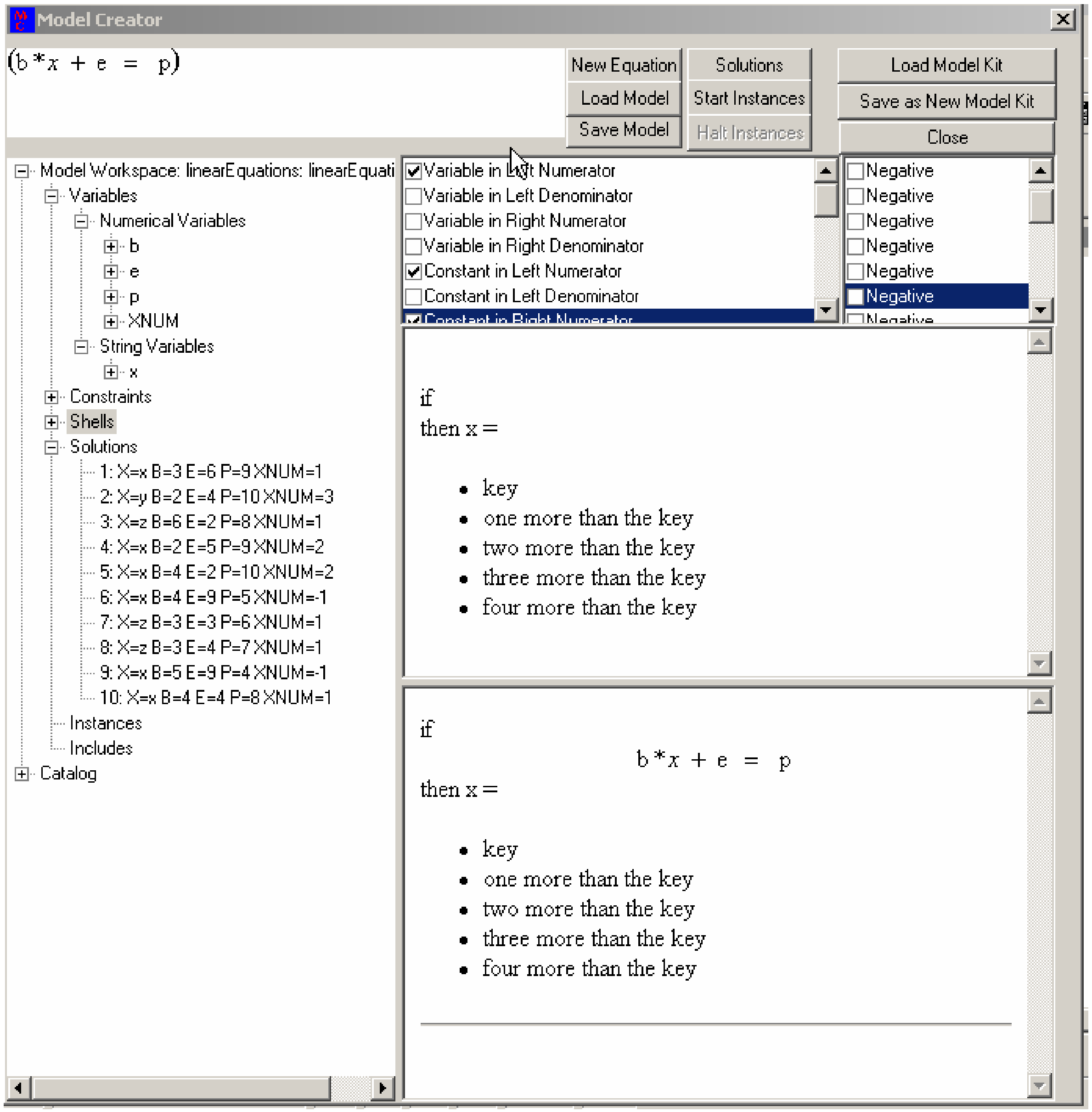
then x =

- key
- one more than the key
- two more than the key
- three more than the key
- four more than the key

***Figure 10.* Model Creator after the generation of equation variants.**

Because the act of creating an equation also created the constraints, alternate versions of the equation, and variant versions of the shell files, one step remains before actual instances of the model can be calculated: Solutions have to be calculated. Clicking on the *solutions* button brings up a menu asking how many solutions will be made available, and when they are calculated, they are displayed under their own heading in the left-hand pane. See Figure 11.

The solutions are listed in this diagram as specific combinations of values for each of the variables. They are consistent with the range of values specified in earlier screens, but have been narrowed down to single combinations of choices as they represent the final step before creation of instances.



```
(b*x + e = p)
```

```
Model Workspace: linearEquations: linearEquati
  Variables
    Numerical Variables
      b
      e
      p
      XNUM
    String Variables
      x
  Constraints
  Shells
  Solutions
    1: X=x B=3 E=6 P=9 XNUM=1
    2: X=y B=2 E=4 P=10 XNUM=3
    3: X=z B=6 E=2 P=8 XNUM=1
    4: X=x B=2 E=5 P=9 XNUM=2
    5: X=x B=4 E=2 P=10 XNUM=2
    6: X=x B=4 E=9 P=5 XNUM=-1
    7: X=z B=3 E=3 P=6 XNUM=1
    8: X=z B=3 E=4 P=7 XNUM=1
    9: X=x B=5 E=9 P=4 XNUM=-1
    10: X=x B=4 E=4 P=8 XNUM=1
  Instances
  Includes
Catalog
```

- [✓] Variable in Left Numerator
- [ ] Variable in Left Denominator
- [ ] Variable in Right Numerator
- [ ] Variable in Right Denominator
- [✓] Constant in Left Numerator
- [ ] Constant in Left Denominator
- [✓] Constant in Right Numerator

- [ ] Negative
- [ ] Negative
- [ ] Negative
- [ ] Negative
- [ ] Negative
- [ ] Negative
- [ ] Negative

if
then x =

- key
- one more than the key
- two more than the key
- three more than the key
- four more than the key

if

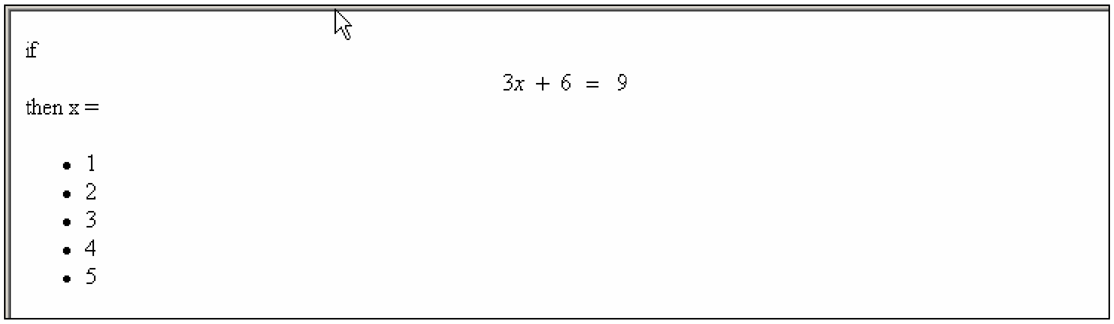
$$b*x + e = p$$

then x =

- key
- one more than the key
- two more than the key
- three more than the key
- four more than the key

***Figure 11*. Model Creator after the generation of solutions.**

At this point, instances of the model can be created by clicking the *start instances* button. The resulting file is based upon the *shell-variant file* with solution values substituted for variable elements in the file (Figure 12).


if

$$3x + 6 = 9$$

then x =

- 1
- 2
- 3
- 4
- 5

Figure 12. **Instance generated from a model.**

This example of item-model creation and item generation is oversimplified as an actual linear-equation item, but it illustrates the workings of several crucial parts of the prototype. The choice of equation form, shell, equation variant, and solution set represents decisions a test developer might make at a design level, and these elements would be resources that form part of the model grid for this item type. What is displayed to the test developer is an interface in which these choices are explicit (note that the interface allows the user to eliminate particular shells, particular equation variants, or particular solutions from the final model). But under the surface there is a detailed structure that reflects a (partial) model-grid analysis of the tasks in question, in this case Type 1A linear/simple rational equation items. The current Model Creator interface has been extended to model nearly all the Type 1A linear-equation types (with the exception of the verbal stem items, which would require an additional NLG module) and has been applied to examples of all of the Type A classes, though not to the Type B classes, as these would require obtaining or constructing a module for converting mathematical formulae to text in accordance with ETS guidelines. It has also been applied on an experimental basis to existing NLG-based item classes, such as distance-rate-time word problems, and to a number of other algebra item classes, including linear inequalities, exponents, fractions, and percents.

Behind the Interface: Elements of the Model Kit

The prototype interface displayed above is designed to be used by test developers during item development. But before test developers can use such an interface to design item models, an important first step must be taken: the development of the model kit in consultation with experienced test developers. In the linear-equation work described in this paper, this aspect of the work was first undertaken by developing a set of item models using Math TCA and then analyzing the resulting models for common elements and variations. It is possible that future work will require very different tools to support model analysis. In either case, though, a model kit must be prepared that contains the resources needed to create item models for a particular task.

The most important aspect of a model kit file for quantitative items is a specification of the broad mathematical content—for example, the class of equations to which a particular model kit can be applied. The current prototype handles this specification in part by associating a model kit with a *master equation*, that is, an equation that: (a) contains all of the mathematical structure and information necessary to handle the most complex structural cases, and (b) can be simplified to produce more ordinary cases. The basic strategy followed in producing master equations is that they must contain the full range of variation observed in actual items. The result when a model kit is written is typically fairly complex. For instance, the most complex form for the master equation in the sample model kit can be expressed as:

$$\frac{a*(b*x + c*x + d) + e}{f*(g*x + h*x + i) + j} = \frac{k*(l*x + m*x + n) + p}{q*(r*x + s*x + t) + u}.$$

Each piece of the form represents some combination of elements that appear across a wide range of actual linear-equation items. For instance, some equations have two instances of the variable as added terms, others have constants inside or outside a parenthesis, others use material in the denominator, etc. The decision to include particular elements in the master equation is a pragmatic one that requires a balance between ease of use and the completeness with which varying forms can be generated from a single model kit. It embodies an analysis of what features of form distinguish variant equations. No one is likely to use all of these elements in any single item, but any given equation form can be produced by selecting only those features present in the equation form desired.

The maximal equation one gets by selecting all possible features is much more complex than a standard linear equation, and, in fact, need not simplify to a linear equation at all. But it carries within it most of the major variations in form that can be observed in actual linear equations. To get an actual linear equation out of the master equation, it is necessary only to set parts of the equation to 1 or 0 and to simplify the equation, leaving out negation if terms are positive. For instance, a simple linear equation of the form $b * x + e = p$ can be derived by setting all terms but *b*, *e,* and *p* to 0 or 1 as appropriate, specifying none of them as negative, and running an equation-simplifying routine.

The master equation is stored in MathML format and comes packaged with two menu-choice files that specify how particular menu choices correspond to modifications of the master equation. For instance, Table 4 shows one of the menu specifications.

Table 4

Modifications of the Master Equation

Variable referenced in the master equation	Value when not selected	Text to be displayed in the user interface
b	0	Variable_in_Left_Numerator
g	0	Variable_in_Left_Denominator
l	0	Variable_in_Right_Numerator
r	0	Variable_in_Right_Denominator
e	0	Constant_in_Left_Numerator
j	0	Constant_in_Left_Denominator
p	0	Constant_in_Right_Numerator
u	0	Constant_in_Right_Denominator
s	0	Second_Variable_in_Right_Denominator
m	0	Second_Variable_in_Right_Numerator
h	0	Second_Variable_in_Left_Denominator
c	0	Second_Variable_in_Left_Numerator
t	0	Summation_of_Constants_in_Right_Denominator
n	0	Summation_of_Constants_in_Right_Numerator
i	0	Summation_of_Constants_in_Left_Denominator
d	0	Summation_of_Constants_in_Left_Numerator
q	1	Distributed_Constant_in_Right_Denominator
k	1	Distributed_Constant_in_Right_Numerator
f	1	Distributed_Constant_in_Left_Denominator
a	1	Distributed_Constant_in_Left_Numerator

This specification indicates the relationship between the menu displayed to the user and the modifications that will be made to the master equation. The first column indicates which variables in the master equation are affected. The second column indicates what will happen when a menu choice is not selected. For instance, if we have the form $q * (r * x + s * x + t) + u$, and q is not selected, it will be assigned the value 1, which will cause it to be removed from the equation as a redundant element. Similarly, if s and t are not selected, they will be set to 0, which will cause the terms that contain them to also be removed, yielding a simplified form such as $rx + u$. The third column contains the text that will be displayed to the user when he or she is deciding which features to select.

In Table 5, a second specification gives modifications to the terms themselves, in this case, by adding the possibility of negation.

Once again, the first column indicates the term to modify, the second column indicates the value to substitute for it, and the third column indicates the information to put in the menu. In this case, all that is being controlled is the presence or absence of negation, which will translate as appropriate into subtraction or a negative term.

Critically, the master equation and the menu choices are defined as data, so that the system can be extended to cover new equation types and new modifications of the base equation form simply by changing the textual data that defines them.

The master equation plus the menu choices are sufficient to define a family of equation forms; a text-processing module takes the master equation plus the actual choices from menu selections and derives an actual equation form for use in the rest of the system. This equation form is used directly to construct mathematical constraints, and also drives any direct display of the underlying equations in an actual text presentation of an item. In particular, there is a module that produces algebraically equivalent equations from a base equation form, which derives (among others) the following variants of the equation form $b*x+e=p$:

$$p = bx + e \qquad e = -bx + p \qquad -bx + p = e \qquad bx = -e + p \qquad -e + p = bx$$

$$e = p - bx \qquad p - bx = e \qquad bx = p - e \qquad p - e = bx \qquad -p + bx + e = 0$$

Now, if deriving a base equation form is one dimension of the model grid, and deriving the possible modifications of that form is a second, then a third dimension is the choice of a model shell.

Table 5

Second Set of Modifications to the Master Equation

Variable referenced in the master equation	Value assigned to variable when negative is not selected	Text displayed in the user interface
b	-b	Negative
g	-g	Negative
l	-l	Negative
r	-r	Negative
e	-e	Negative
j	-j	Negative
p	-p	Negative
u	-u	Negative
s	-s	Negative
m	-m	Negative
h	-h	Negative
c	-c	Negative
t	-t	Negative
n	-n	Negative
i	-i	Negative
d	-d	Negative
q	-q	Negative
k	-k	Negative
f	-f	Negative
a	-a	Negative

In the simplest case, a shell corresponds essentially to a choice of presentation mode for the data. The Model Creator interface is designed to work seamlessly with shell files in HTML or XML format, including an XML file format for multimedia presentations using the Media Semantics Character Toolkit (http://www.mediasemantics.com). It thus demonstrates a broad range of ways in which the content from a model could be prepared for presentation and routed through a presentation model.

In the current implementation, shells are treated essentially as templates into which information is substituted, except that the resulting file is potentially processed further by a presentation module.

The prototype user interface interprets special tags and expressions as providing information about slots in which information will be substituted, including:

1. The ability to substitute an entire equation into the shell
2. The ability to substitute the left- or right-hand side of an equation into the shell
3. The ability to define a substitution point for a variable
4. The ability to embed complex expressions inside a shell file using functions

The current prototype is able to evaluate any combination of functions defined in the constraint engine also used in the Math TCA interface.

There is an important issue that explains the decision to allow fairly rich behind-the-scenes structure in shells. Ideally, the components of a model analysis should document themselves; that is, it should be possible for a test developer to understand what a model will do from the component pieces without having to understand the details of how it has been programmed. On the other hand, the person who creates a model kit needs to have fine-grained control over what happens both in the variables and constraints that define a model and in the way this gets expressed in item shells.

Based on this philosophy, a number of other features are included in the current prototype:

1. The ability to include files
2. The ability to select random combinations of items from a list
3. The ability to require that particular variables be defined or not defined for a shell to be available for a particular model

The key idea is that much of the information that is associated with the model itself in the Math TCA interface is a property of a particular shell template and should be handled as such.

A final aspect of the prototype is a replication of the ability to do low-level, detailed item modeling using individually defined variables and constraints, replicating most of the functionality supported by the Math TCA interface.

Conclusions and Prospects for Future Work

The approach embodied in the prototype critically assumes that much of the variation in the model grid for quantitative items reflects common dimensions of numerical and computational complexity and thus that a single tool can support analysis of the model grid across a wide range of item types. We have identified quantitative reasoning variables that are not specific to particular equation forms but that also appear related to difficulty. Additional work needs to be undertaken to confirm the usefulness of this approach over a broader range of item types.

Note that the current prototype depends critically upon the creation of resources—model kits—embodying major structural variables within a content domain, and that this prototype can be extended to new task types by creating model kits containing new master equations, menus for equation variation, and shell types. Planned modifications of the prototype will take these dimensions into account to support maximally flexible creation of model kits. In addition, the prototype needs to be integrated more closely with previous work on word problems, and most importantly, the entire framework sketched here. Integrating ECD with item modeling presupposes a concerted effort to relate the task grid and model grid to difficulty. The most immediate issues that need to be addressed in this regard are variation in numerical complexity and analysis of the role of distractors.

Numerical Complexity

As mentioned earlier, it is well-known that certain kinds of numbers are simpler than others and are mastered at an earlier age: whole numbers before rational numbers, mathematical operations on fractions with like denominators before similar operations on fractions with different denominators, operations on single-digit numbers before operations on multiple-digit numbers, and so on. Ideally, test developers should not have to specify the characteristics of variables in a model in detail. They should be able to make design level decisions—such as "variables are single digit whole numbers," "variables are decimals to three places," or "variables are fractions with like denominators"—arranged systematically from the simplest to the most complex cases. Creating an appropriate grid for numerical complexity will be a critical element in future work.

The Role of Distractors

As Minstrell and Hunt (2004) and Thissen-Roe and Hunt (2004) argue and as discussed earlier, distractors in a multiple-choice item can be submitted to systematic analysis and treated as reflecting facets of student (mis)understanding of the target domain. In the current prototype, distractors are handled as part of the item shell. Given the very promising results of distractor analysis, it seems probable that the performance of a system like the current prototype can be improved by separating out distractor analysis and creating a separate component that generates plausible distractors and determines how they can reasonably be combined. Creating a general facility for managing facets and the distractor patterns associated with them remains a critical element for future work.

Word Problems

The current prototype has been extended to integrate the work on word problems reported in Deane (2003) and Deane and Sheehan (2003). The XML document specification used by the NLG system is treated as an item shell, and the (NLG components are treated as presentation modules that take the document specification and render it in English (or another language, in particular Japanese and Spanish, as in Higgins et al., 2005). Variations in wording are treated as shell variations generated by the system automatically combining alternative XML specifications. One advantage of the shell-based approach in the current prototype is that the system can be designed flexibly to include a range of NLG approaches, from simple template-based approaches (for items like linear/simple rational equations, where the variations in wording are minimal) to the full NLG system (appropriate for complex word problems, such as distance-rate-time problems).

Difficulty Modeling

The current prototype can best be viewed as projective in its approach to difficulty modeling: Its goal has been to identify the dimensions of variation that seem potentially relevant as features in a task-difficulty model. It does not in and of itself provide evidence about the actual structure of a difficulty model. An important next step should be to use this framework as a springboard into the construction of task-difficulty models in which the factors that enter into the ECD analysis and the features of the difficulty model coincide.

Prospectus

Critical to the approach outlined here is the concept of a two-level task analysis, in which a relatively abstract task grid is developed as part of the ECD analysis, followed by model-grid analysis for particular cells of the task grid. If the types of variation within a model grid can be shown to be relatively consistent across tasks—that is, if the kinds of variation can be reduced to a relatively small number of types, of the sort outlined above—then the task of developing a tool to support model creation for particular tasks could be simplified considerably, particularly if it can be directly linked to a difficulty model. The feasibility of such an approach requires, however, a more general study of many different item types, and such a general study is critical to determining the success and direction of future work on the relationship between item modeling and ECD.

The overall direction discussed here is based upon a number of guiding principles that we believe will be helpful in moving toward an approach in which item modeling and task modeling are closely integrated within an ECD framework. The following points should be noted in particular:

1. As matters currently stand, item models can easily be written in such a way that easier and simpler versions of an item can be produced by accident.
2. One reason for this state of affairs is the fact that there is no explicit connection made between the specification of an item model and those task-model variables known to control difficulty.

The current prototype software described in this report represents an effort to address these issues and to enable more explicit software support for a workflow in which item modeling and task modeling are part of the same process.

References

Bejar, I. I. (1990). A generative analysis of a three-dimensional spatial task. *Applied Psychological Measurement, 14*(3), 237-245.

Bejar, I. I. (1993). A generative approach to psychological and educational measurement. In N. Frederiksen, R. J. Mislevy & I. I. Bejar, (Eds.), *Test theory for a new generation of tests* (pp. 323–359). Hillsdale, NJ: Lawrence Erlbaum Associates.

Bejar, I .I. (1996). *Generative response modeling: Levering the computer as a test delivery medium* (ETS RR-96-13). Princeton, NJ: ETS.

Bejar, I .I. (2002). Generative testing: From conception to implementation. In S. H. Irvine & P. Kyllonen (Eds.), *Item generation for test development* (pp. 199–218). Mahwah, NJ: Lawrence Erlbaum Associates.

Bejar, I .I., Lawless, R., Morley, M. E., Wagner, M. E., Bennett, R. E., & Revuelta, J. (2002). *A feasibility study of on-the-fly item generation in adaptive testing* (GRE Board Professional Report No. 98-12P; ETS RR-02-23). Princeton, NJ: ETS.

Bejar, I. I., & Yocom, P. (1991). A generative approach to the modeling of isomorphic hidden-figure items. *Applied Psychological Measurement, 15*(2), 129–137.

Borchert, K. (2000). *Connecting operational and structural knowledge in algebra: The impact of word problem solving on equation construction.* Unpublished master's thesis, University of Washington.

Borchert, K. (2003). *Disassociation between arithmetic and algebraic knowledge in mathematical modeling.* Unpublished doctoral dissertation, University of Washington.

Cahill, L., Doran, C., Evans, R., Mellish, C., Paiva, D., Reape, M., et al. (1999). In search of a reference architecture for NLG systems. In *Proceedings of the 7th European Workshop on Natural Language Generation (EWNLG '99*'; pp. 77–85).

Clement, J., Lochhead, J., & Monk, G. S. (1981). Translation difficulties in learning mathematics. *American Mathematical Monthly, 88*(4), 286–290.

Clement, J., Lochhead, J., & Soloway, E. (1979). *Translating between symbol systems: Isolating a common difficulty in solving Algebra word problems*. Amherst, MA: University of Massachusetts.

Deane, P. (2003). *The Model Creator program for natural language generation of math word problems* (Internal Report). Princeton, NJ: ETS.

Deane, P., & Sheehan, K. (2003, April). *Automatic item generation via frame semantics: Natural language generation of math word problems*. Paper presented at the annual meeting of the National Council on Measurement in Education, Chicago.

Embretson, S. E. (1994). Application of cognitive design systems to test development. In C. R. Reynolds (Ed.), *Cognitive assessment: A multidisciplinary perspective* (pp. 107-135). New York: Plenum Press.

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

Embretson, S. E., & Schneider, L. M. (1989). Cognitive models of analogical reasoning for psychometric tasks. *Learning and Individual Differences, 1*, 155–178.

Embretson, S. E., & Waxman, M. (1989, November). *Models for processing and individual differences in spatial folding.* Paper presented at the meeting of the Psychonomic Society, St. Louis, MO.

Enright, M. K., Morley, M., & Sheehan, K. M. (2002). Items by design: The impact of systematic feature variation on item statistical characteristics. *Applied Measurement in Education 15*(1), 49-74.

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.

ETS. (1996). *GRE big book: Practicing to take the general test*. Princeton, N.J: Author.

Fillmore, C. J. (1968). The case for case. In E. Bach & R. T. Harms (Eds.), *Universals in linguistic theory* (pp. 1–88). New York: Holt, Rinehart, and Winston.

Fillmore, C. J. (1976). Frame semantics and the nature of language. In S. R. Harnad, H. D. Steklis, & J. Lancaster, *Annals of the New York Academy of Sciences: Vol. 289. Conference on the origin and development of language and speech* (pp. 20-32). New York: New York Academy of Sciences.

Fillmore, C. J. (1977a). Scenes-and-frames semantics. In A. Zampolli (Ed.), *Fundamental studies in computer science: Vol. 59. Linguistic Structures Processing* (pp. 55-88). Amsterdam: North Holland Publishing.

Fillmore, C. J. (1977b). The need for a frame semantics in linguistics. In H. Karlgren (Ed.), *Statistical methods in linguistics.* Stockholm: Skriptor.

Fillmore, C. J. (1982): Frame semantics. In Linguistic Society of Korea (Ed.), *Linguistics in the morning calm* (pp. 111–137).Seoul, South Korea: Hanshin Publishing Co.

Fillmore, C. J. (1985). Frames and the semantics of understanding. *Quaderni di Semantica*, *6*(2), 222–254.

Fillmore, C. J., & Atkins, B. T. S. (1994). Starting where the dictionaries stop: The challenge for computational lexicography. In B. T. S. Atkins & A. Zampolli (Eds.), *Computational approaches to the lexicon*. Oxford, England: Clarendon Press.

Fillmore, C. K., & Baker, C. F. (2001). Frame semantics for text understanding. In *Proceedings of the NAACL 2001 Workshop WordNet and Other Lexical Resources: Applications, Extensions and Customizations*.

Goldin, G. A., & McClintock, C. E. (Eds.). (1984). *Task variables in mathematical problem solving.* Philadelphia: Franklin Institute Press.

Graf, E. A., Bassok, M., Hunt, E., & Minstrell, J. (2004). A computer-based tutorial for algebraic representation: The effects of scaffolding on performance during the tutorial and on a transfer task. *Technology, Instruction, Cognition, and Learning, 2*(1-2), 135–170.

Graf, E. A., Peterson, S., Steffen, M., & Lawless, R. R. (2005). *Psychometric and cognitive analysis as a basis for the design and revision of quantitative item models* (ETS RR-05-25). Princeton, NJ: ETS.

Graf, E. A., & Shute, V. (2004). *Using an item modeling approach to develop mathematics assessment items for an 8th-grade unit on sequences.* Manuscript in preparation.

Harel, G., & Behr, M. (1989). Structure and hierarchy of missing value proportion problems and their representations. *Journal of Mathematical Behavior, 8*, 77–119.

Harel, G., Behr, M. J., Post, T., & Lesh, R. (1991). Variables affecting proportionality: Understanding of physical principles, formation of quantitative relations and multiplicative invariance. In F. Furinghetti (Ed.), *Proceedings of PME XV Conference* (pp. 125-133). Assisi, Italy: PME.

Harel, G., Behr, M., Post, T., & Lesh, R. (1994). The impact of the number type on the solution of multiplication and division problems: Further considerations. In G. Harel & J. Confrey (Eds.), *The development of multiplicative reasoning in the learning of mathematics* (pp. 363–384). Albany, NY: State University of New York.

Higgins, D., Futagi, Y., & Deane, P. (2005). *Multilingual generalization of the Model Creator software for Math TCA* (ETS RR-05-02). Princeton, NJ: ETS.

Hively, W., Patterson, J. L., & Page, S. H. (1968). A “universe-defined” system of arithmetic achievement tests. *Journal of Educational Measurement, 5*(4), 275–290.

Hunt, E. B., & Minstrell, J. (1994). A cognitive approach to the teaching of physics. In K. McGilly (Ed.), *Classroom lessons: Integrating cognitive theory and classroom practice* (pp. 51–74). Cambridge, MA: MIT Press/Bradford Books.

Hunt, E. B., & Minstrell, J. (2003). *Building a facet-based, diagnostic assessment system for improving science and mathematics learning in classrooms*. Retrieved March 10, 2006, from http://www.Diagnoser.com

Johnson, M. S., & Sinharay, S. (2003, August). *Calibration of polytomous item families using Bayesian hierarchical modeling* (ETS RR-03-23). Princeton, NJ: ETS.

Koedinger, K. R., & Nathan, M. J. (2004). The real story behind story problems: Effects of representations on quantitative reasoning. *Journal of the Learning Sciences, 13*(2), 129–164.

Kyllonen, P. (2002). Item generation for repeated testing of human performance. In S. Irvine & P. Kyllonen (Eds.), *Item generation for test development* (pp. 251–276). Mahwah, NJ: Lawrence Erlbaum Associates.

Lewis, A. B., & Mayer, R. E. (1987). Students' miscomprehension of relational statements in arithmetic word problems. *Journal of Educational Psychology, 79*(4), 363–371.

Marshall, S. (1995). *Schemas in problem solving*. New York: Cambridge University Press.

Martin, S. A., & Bassok, M. (2005). Effects of semantic cues on mathematical modeling: evidence from word-problem solving and equation construction tasks. *Memory and Cognition, 33*(3), 471-478.

Mayer, R. E. (1982a). Different problem-solving strategies for algebra word and equation problems. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 8*(5), 448–462.

Mayer, R. E. (1982b). Memory for algebra story problems. *Journal of Educational Psychology, 74*(2), 199–216.

Meisner, R. M. Luecht, R., & Reckase, M. D. (1993). *The comparability of the statistical characteristics of test items generated by computer algorithms* (ACT Research Report Series No 93-9). Iowa City, IA: The American College Testing Program.

Minstrell, J., & Hunt, E. B. (2004, June). Item modeling and facet-based assessment. In R. R. Lawless (Chair), *Item modeling and facet-based assessment in math and science: Applications to high-stakes and formative assessments*. Symposium conducted at the 30th annual conference of the International Association for Educational Assessment (IAEA), Philadelphia, PA.

Mislevy, R. J., Almond, R. G., & Lukas, J. F. (2003). *A brief introduction to evidence-centered design* (ETS RR-03-16). Princeton, NJ: ETS.

Mislevy, R. J., & Riconscente, M. M. (2005). *Evidence-centered assessment design: Layers, structures, and terminology* (PADI Tech. Rep. No. 9). Retrieved May 25, 2006, from http://padi.sri.com/downloads/TR9_ECD.pdf

Mislevy, R. J., Steinberg, L. S., & Almond, R. G. (2003). On the structure of educational assessments. *Measurement: Interdisciplinary Research and Perspectives, 1,* 3-67.

National Assessment Governing Board. (2001). *2005 NAEP mathematics assessment and item specifications* (pre-publication edition). Washington, DC: U.S. Department of Education.

Payne, S. J., & Squibb, H. R. (1990). Algebra mal-rules and cognitive accounts of error. *Cognitive Science, 14*(3), 641–642.

Reiter, E. (1995). NLG vs. templates. In K. de Smedt, C. Mellish, & H. J. Novak (Eds.), *Proceedings of the Fifth European Workshop on Natural Language Generation* (pp. 95-106). Leiden, The Netherlands: Rijks Universiteit Leiden.

Riley, M. S., Greeno, J. G., & Heller, J. I. (1983). Development of children's problem-solving ability in arithmetic. In H. P. Ginsburg (Ed.), *The development of mathematical thinking* (pp. 153–196). New York: Academic Press.

Sebrechts, M. M., Enright, M., Bennett, R. E., & Martin, K. (1996). Using algebra word problems to assess quantitative ability: Attributes, strategies, and errors. *Cognition and Instruction, 14*(3), 285–343.

Sheehan, K., & Kostin, I. (2006). *A cognitive linguistic analysis of the skills underlying performance on GRE sentence completion items*. Manuscript in preparation.

Shute, V. J., Graf, E. A., & Hansen, E. (2005). Designing adaptive, diagnostic math assessments for sighted and visually disabled students. In L. PytlikZillig, R. Bruning, & M. Bodvarsson (Eds.), *Technology-based education: Bringing researchers and practitioners together.* Greenwich, CT: Information Age Publishing.

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*. Mahwah, NJ: Lawrence Erlbaum Associates.

Sinharay, S., Johnson, M. S., & Williamson, D. (2003, February). *An application of a Bayesian hierarchical model for item family calibration* (ETS RR-03-04). Princeton, NJ: ETS.

Sinharay, S., & Johnson, M. (2005). *Analysis of data from an admissions test with item models* (ETS RR-05-06). Princeton NJ: ETS.

Sleeman, D. (1984). An attempt to understand students' understanding of basic algebra. *Cognitive Science, 8,* 387-–412.

Steinberg, L., Mislevy, R. J., & Almond, R. G. (2005). *U.S. Patent No. 434350000.* Washington, DC: U.S. Patent and Trademark Office.

Thissen-Roe, A., & Hunt, E. B. (2004, June). A scaling technique for analyzing data from formative testing. In R. R. Lawless (Chair), *Item modeling and facet-based assessment in math and science: Applications to high-stakes and formative assessments.* Symposium conducted at the 30th annual conference of the International Association for Educational Assessment (IAEA), Philadelphia, PA.

Whitely, S. E., & Schneider, L. M. (1981). Information structure on geometric analogies: A test theory approach. *Applied Psychological Measurement, 5,* 383–397.

Notes

[1] Henceforth, when we refer to *tasks*, we are referring to the underlying processes that examinees must execute in order to reach a correct answer in an item. In an evidence-centered design approach (Mislevy & Riconscente, 2005; Mislevy, Steinberg, & Almond, 2003), a task model provides a framework for describing and constructing situations in which examinees act. Within each task model are task-model variables that include specifications for the stimulus material, conditions necessary to produce the desired behaviors, and the work products that the examinees will produce in response to the tasks, that is, the format of the item as well as the format of the expected response.

[2] A *string variable* is defined as a variable whose representation is alphabetic, that is, words, phrases, or proper names.

[3] “Other recursive” refers to sequences commonly encountered in high school algebra that have a readily identifiable recursive rule and are neither arithmetic nor geometric. These sequences may also be described by an explicit rule, though often this is more difficult and may or may not be a task requirement (see point 3 in representation of rule for sequence).

[4] *GRE Big Book* (ETS, 1996), Test 9, Section 5, p. 373, Item 16.

[5] *GRE Big Book* (ETS, 1996), Test 18, Section 6, p. 857, Item 16.

[6] *GRE Big Book* (ETS, 1996), Test 11, Section 1, p. 425, Item 20. It should be noted that the decision to code an item with a verbal representation as “1-equation” or “2-equation” is a judgment call; the examinee might represent it either way, if, in fact, an algebraic representation is used at all. Here we try to ascertain whether the text of the stem more strongly suggests a system of two equations or a single equation.

[7] Four of the 62 items from the initial set were excluded from consideration because they did not fit into this grid. Two of these items were algebraic representation tasks; the stem described a mathematical relationship in verbal form and the examinee had to select an equation that correctly translated the verbal description into symbolic form.

[8] We did not have any reason to expect effects due to interactions among the attributes, but we did compare a regression model that entered main effects only with a regression model that

entered both main effects and first order interaction effects. The change in R^2 as a result of adding the interaction effects was not significant.