University of Alberta

Developing a Framework and Demonstrating a Systematic Process for Generating Medical Test Items

by

Hollis Lai

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Measurement, Evaluation and Cognition

Department of Educational Psychology

Spring 2013
Edmonton, Alberta

Abstract

Automatic item generation (AIG) is a field of research dedicated to the production of test items using computer technology. Despite dramatic developments on AIG in the past decade, there is little documentation on a general methodology for creating the models needed to generate items. The purpose of my dissertation is to introduce a three stage framework to guide the process of item generation and to present a proof-of-concept application demonstrating how items can be generated using this framework. The generative framework involves individual stages of development: cognitive modeling, item modeling and item generation. My unique contribution to the literature is threefold. First, I present a modeling approach for extracting knowledge from content experts that can be used for item generation. Second, I present a template-based item generation technique named n-layer modeling to minimize text similarity between generated items from the same model. Third, I present a method for evaluating text similarity for generated items. These methods and procedures are demonstrated in the context of medical education, specifically in the area of surgery. Results generated test items in the domain of hernia and post-operative fever. The application of n-layer item modeling demonstrated a generation technique that can produce more test items, and test items with less text similarity. By integrating test development processes with technology, the generation framework proposed in this study can combine a systematic and iterative process with the humanistic task of providing content

expert knowledge to produce test items en masse for meeting current educational testing demands.

Acknowledgements

There are many whom I owe sincere thanks for contributing to my learning. For sharing your time, insightful conversations, and interdisciplinary guidance, thank you Drs. Buck, Carbonaro and Stroulia for guiding me throughout the dissertation process and serving on my committee. I would like to thank Dr. Eva for participating as my external examiner and providing me with detailed insights for improving my study.

For teaching and guiding me with kind patience and understanding, thank you Drs. Rogers, Leighton and Cui. Your depth of knowledge, critical thinking, and clear explanation to complex ideas for the past five years have inspired and formed my way of thinking. For your kind collaboration and insightful discussions, thank you Drs. Babenko, Turner, Alves and all CRAMERs. Your company throughout this period has made the experience memorable. For opening my eyes to the real world of psychometrics, thank you Drs. Becker, Barton and Breithaupt. The practical experiences you have provided me solidified the skills and knowledge I now use every day.

For bearing with me through all my moments, I am thankful to have my parents, Xian, and my friends to share my ups and downs with. For mentoring, guiding, encouraging and just about everything else throughout these years, I cannot thank Dr. Gierl enough for all he has provided me.

Table of Contents

List of Figures

List of Tables

Chapter I: Educational Testing in the 21st Century

Dramatic changes are shaping how we design and administer educational tests. Large-scale tests, once intended exclusively to meet accountability and summative evaluation purposes, are now expected to also provide valuable formative information to guide teaching and promote learning (Ferrara & De Mauro, 2006). To meet these teaching and learning directives, formative assessment principles are beginning to guide our educational testing practices. Formative principles include any assessment-related activities, such as administering tests more frequently, that yield constant and specific feedback to modify teaching and improve learning. But when testing occurs frequently, more tests are required and these tests must be created both efficiently and economically. With more frequent testing, exams no longer serve as an intermittent "rite of passage", but are now considered to be an integral part of the teaching and learning process.

Because testing occurs more frequently, more tests are required. Tests that enhance instruction require more test forms and, ultimately, more test items. Unfortunately, due to our current debt-laden economic climate, extensive cuts to public education have persuaded educators that they must achieve more with less (Camara, 2011). To maintain or even improve the quality of testing without increasing costs, testing programs must operate in a more efficient and cost-conscious manner. This seemingly contradictory idea—educational testing occurring more frequently, satisfying more purposes, and

providing more information to students and teachers, but requiring the same, or, even less than, the cost of existing testing programs—is now a serious question that must be addressed by educators who develop 21st century educational assessments. Fortunately, the demand for more timely and frequent educational testing coincides with a rise in the application and use of educational technology. Computer and technology use has changed the way we live, learn, and interact with others. Bennett (2001) predicted that no topic would become more central to innovation and future practice in educational testing than computers and the internet. One decade after this bold but accurate claim, we are beginning to see two waves of change, enabled by technology, that affect the way we design and deliver educational tests.

Assessment and Technology

The first wave of technological change in educational testing improves how we deliver tests. Large-scale tests are no longer feasible when delivered in a paper-based format. The printing, scoring, and reporting of paper-based tests require tremendous efforts, expenses, and human resources (Parshall, 2002). Moreover, as the demand for more frequent testing escalates, the cost of administering paper-based tests will also increase, likely in direct proportion to the number of tests required. One solution that can help curtail test administration cost is to transfer to a computer-based testing (CBT) system. By administering tests on computers over the Internet, students no longer have to rely on their favourite "HB" pencil to answer each item. More importantly,

educators are liberated from performing the time-consuming administration and scoring processes associated with disseminating, scanning, and scoring paper-based tests. Instead, tests can be administered by computers on-demand, scored automatically, thereby allowing students to receive immediate feedback on their performance. Costs accrued from tasks such as printing and scanning test forms can now be shifted to improve other test-related activities, such as item development (Drasgow, 2002). CBT also allows assessment specialists to implement new test designs such as adaptive testing (Weiss, 1982; Folk & Smith, 2002) where items are administered to examinees based on their ability estimates, thereby allowing educators to meet different demands posed by a diverse set of testing purposes, but with less testing time and at lower test development and administration costs. With more than 27 of the 50 US states now administering computer-based educational tests (as cited in Gierl & Lai, 2011), and a large number of high profile exams such as the Graduate Records Exam (GRE), Graduate Management Achievement Test (GMAT), American Institute of Certified Public Accountants Uniform CPA examination (CBT-e), and National Council of State Boards in Nursing (NCLEX) administering exams electronically, computer-based testing has become standard practice.

The second wave of change made possible through the use of computer-based testing is in the area of test design. Many examples of innovative test designs can be cited. For example, tests can now be used to identify students' cognitive-problem solving strengths and weaknesses as well as yielding

diagnostic information on these specific areas (Leighton & Gierl, 2007a; Huff & Goodman, 2007). Tests can be designed to track the learning progress of students in a particular subject over time using multiple test administrations (Liu, 2010). Tests contain innovative item types that prompt student responses from a variety of stimuli thereby allowing educators to measure more complex performances as well as a broader variety of knowledge, skills, and competencies (Scalise & Gifford, 2006). To implement these new and ambitious test design approaches, expert collaboration across disciplines such as educational psychology, computing science, mathematical statistics, educational technology, and cognitive psychology is required (Gierl & Lai, 2011). In sum, diverse and dramatic advancements in educational testing today are all made possible by the emergence and use of technology.

The transition to computer-based testing coupled with advances in test design permit educators to evaluate students more frequently (Drasgow, 2002). But the desirability to test students more frequently combined with the need to test for many different purposes has also exposed one fatal constraint with our current educational testing paradigm—the creation of new test items (Downing & Haladyna, 2006; Schmeiser & Welch, 2006). Computer-based tests must be supported by a large bank of items because frequent testing means that items are continuously administered and, therefore, exposed to students. A bank is a repository of test items along with their associated data. Data recorded for each item include content and psychometric characteristics (e.g., difficulty level of the

item), which is required for test assembly, as well as usage (e.g., exposure rate), which is required for test security. These banks must be frequently replenished with new test items to ensure that students receive a continuous supply of unique, content-specific, items while limiting item exposure to maintain test security within the testing environment to ensure fairness for all students. Item exposure represents a serious problem for the integrity of the testing process. Examples of such compromise in a medical licensure (Zamost, Griffin & Ansari, 2012) or student achievement (O'Brien, 2012) context immediately garner international media attention (O'Neil, 2012) and highlight the consequences of test security in a computer-based examination context.

To minimize the risk of item exposure, large item banks must be developed. Breithaupt, Ariel, and Hare (2010), for example, estimated that the number of items required for a 40-item adaptive test with two administrations per year was, at minimum, 2,000 items. A much smaller number of items is needed for a paper-based test, largely, because these tests are only administered in fixed length forms (i.e., test was not adaptive) and with a small number of administrations per year (e.g., one administration every 12 months). Modern educational tests, by way of contrast, are administered multiple times per year using a design where the test provides detailed information on student performance. These features of testing frequency combining with detailed feedback within a computer-based context can only be operationalized when thousands of test items are available. In short, recent developments have

produced many exciting changes to how we conceptualize and develop educational tests. But these important developments are also predicated on the assumption that we have large banks of items at our disposal. Unfortunately, these item banks are not, in fact, currently available and the means by which we can create such banks in an efficient and cost-effective manner is unclear. Hence, the purpose of my dissertation is to develop and implement a methodology for generating test items.

Technology can be used to improve our test development practices. Research on automated test assembly demonstrates how items can be selected from banks to create test forms with precise content specifications and psychometric properties (van der Linden, 1998). Computer-adaptive testing demonstrates how the administration of test items can be tailored to the ability of each examinee leading to highly reliable test scores using only a small number of test items (Weiss, 1982). Computerized item banking demonstrates how items are stored, indexed, and accessed in an efficient manner (Flaugher, 1999). Despite these important advances in the use of test items, the process of creating new items using technology has remained, largely, unaddressed in the educational measurement literature. Item development is a costly and time-consuming process—this point cannot be over-emphasized. Test items are currently written by a single content expert and then reviewed, edited, and revised by committees of content experts. Using this content expert approach, the cost of developing a single item is estimated to range from $1,500-$2,000

USD (Rudner, 2010). These cost estimates increase even further when the success of the item development process is considered. Approximately 40% of expertly created items fail to perform as intended during field testing, meaning items must either be re-written entirely or discarded altogether (Haladyna, 1994). If we begin to combine the outcomes of these assertions, then we can conclude a testing program that requires a bank of 2,000 operational items will need to develop 3,334 items to account for the 40% level of attrition expected during item development. Moreover, with development cost ranging from $1,500 to $2,000 per item, the program would need to allocate between $5 to $6.6 million for creating the item bank alone. This estimate is staggering considering item development is only one of many processes in any testing program. The prohibitive cost on item development poses serious challenges for testing programs motivated to implement new designs and administration methods that require large pools of items. In short, the practices we currently use to develop test items simply cannot meet the growing demand for these items (Bejar et al., 2002).

Item Generation

Automatic item generation (AIG; Irvine & Kyllonen, 2002; Drasgow, Luecht, & Bennett, 2006; Embretson & Yang, 2007; Gierl & Haladyna, 2012) is a research area dedicated to the production of test items using computer technology. AIG, an idea described by Bormuth (1969) more than 40 years ago, is gaining renewed interest because large numbers of high-quality, content-

specific, test items are needed for the transition to computer-based testing. Generally speaking, AIG can be characterized as the process of using models to generate items with the aid of computer technology. The incorporation of technology as a staple for AIG was discussed by Millman and Westman (1989), when they first proposed item generation strategies contingent on the development of technology. With technology being quite limited in 1989, they developed software to assist with item writing, but they also hypothesized that as technology improved, many more components in the item writing process would be automated. Then, with noteworthy developments from Embretson (1998) on generating items to assess abstract reasoning and Bejar et al. (2002) on developing an "on-the fly" item generation system for computer-based testing, the progression for increased technology-based item development occurred, as Millman and Westman predicted.

To generate items, AIG requires detailed instructions to guide how content should be assembled. These instructions are captured within an item model. An item model is a prototypical representation of a generated item (Bejar, 2002; Gierl & Lai, 2011; LaDuca, Staples, Templeton, & Holzman, 1986), and it serves as a guide to how the item should be structured when completed. Once the item model is specified, the content within the model must be organized and expressed via computer programming. Item generation is the creation of items based on a systematic manipulation of content specified in each item model. By combining item models with item generation, hundreds or

thousands of new items can be produced using a single item model (Gierl, Zhou, & Alves, 2008). The modeling and generating process requires the combination of content expertise and computer technology, as test development specialists identify relevant content and create the models while computers assemble this information to produce new items. Using this process, test developers can create models that yield large numbers of high-quality items in a short period of time.

The transition to computer-based testing combined with the introduction of new test design methods has prompted significant developments in AIG research over the last decade. Important areas of AIG growth include generating items from cognitive models (Embretson & Yang, 2007), item model development (Gierl, Zhou, & Alves, 2008), incorporating item generation into test designs (Huff, Alves, Pellegrino, & Kaliski, 2012; Luecht, 2012), statistical modeling (Embretson, 1999; Geerlings, van der Linden, & Glas, 2011; Glas & van der Linden, 2003; Sinharay, Johnson, & Williamson, 2003), and computer technology (Gierl et al., 2008; Gütl, Lankmayr, Weinhofer, & Höfler, 2011; Higgins, 2007; Higgins, Futagi, & Deane, 2005). Because of these important developments, AIG has been used to create new items in diverse content areas, including but not limited to K-12 levels in subjects such as Language Arts, Social Studies, Science, Mathematics (Gierl et al., 2008) and advanced placement (AP) Biology (Alves, Gierl, & Lai, 2010) as well as psychological domains, including spatial (Bejar, 1990), abstract (Embretson, 2010), figural inductive (Arendasy,

2005), and quantitative reasoning (Arendasy & Sommer, 2007; Embretson & Daniel, 2008; Sinharay & Johnson, 2008), word fluency (Arendasy, Sommer, & Mayr, 2012), visual short-term memory (Hornke, 2002), and mental rotation (Arendasy & Sommer, 2010). In licensure and certification testing, items were generated in areas such as architecture (Gierl et al, 2008), medicine (Gierl & Lai, 2012), nursing (Wendt, Cao, Woo & Bergstrom, 2009), and insurance (Masters, 2010). Despite these important AIG research developments and applications, there is still little information available on a general methodology for creating the models needed to generate items. More specifically, no researcher has yet presented an AIG study that articulates a framework for guiding the process of item generation.

The ambiguity surrounding the item generation process may well be a result of commercial interests, meaning the corporate culture that surrounds testing often leads to the development of proprietary ideas that are not widely shared within the research community. But, I contend, the innovations that currently surround the use of item generation will be fruitless unless a more generalized methodology is developed for use by both researchers and practitioners in the educational testing community. To address this concern, I will develop a framework for guiding item generation and I will illustrate how this framework can be used to create new test items for my dissertation research. There has been some recent attempts to incorporate item modeling into AIG, as a component of a larger assessment design (Bejar,2010). For

instance, Alves, Gierl, and Lai (2010) generated items in the domain of Biology under the evidence-centered design framework (ECD; Mislevy et al., 2003). Lai, Gierl, and Alves (2010) also demonstrated how item generation can be embedded within the assessment engineering framework (AE; Luecht, 2007). While these studies do illustrate how items can be generated from assessment design systems, the specific process of item model development still remains ill-defined. For example, there is no methodology for identifying content elements within the item models or for manipulating this content to produce meaningful items. As researchers in cognitive science recognize the benefits of item modeling and requested more guidelines for the development and use of these models in educational measurement (Leighton & Gokiert, 2005; Bejar, 2010), no study has yet documented a process for item model development.

Demand for Items in Medical Education

No other fields of study could benefit more from item generation than medical education. Canadians are global leaders in medical education for providing innovative and leading edge training to medical professionals (AFMC, 2009). To train, assess, and certify the competencies of future medical professionals, medical education requires testing in many areas of training, administered in a variety of formats and media, with each test associated with a different type of student outcome. As a result, many tests are created to assess student competencies across different levels of outcome, with multiple-choice items as the most common item type. This item type can be used to evaluate

examinees' knowledge and skills across diverse content areas in an efficient and economical manner. However, multiple-choice items are challenging to develop because they require content specialists to combine their interpretations of the professional guidelines and standards of practice for item development with their experiences, expertise, and judgments to structure the knowledge, skills, and content required to solve medical problems in a specific domain.

Medical educators must also attend to, and, eventually respond to the dramatic changes that are now occurring in the fields of educational and licensure testing. All aspects of testing—from the design of assessments to the scoring of items, from the administration procedures to test score reporting—are undergoing profound changes influenced by developments in assessment-related disciplines such as cognitive science, statistics, medical education, educational psychology, operations research, educational technology, and computing science. These changes require tremendous test development efforts that must result in an even greater number of new test items being developed. Coupled with the reality that content specialists for medical education are medical experts, a very constrained group in the population with many other professional tasks and responsibilities, current item writing approaches (see Chapter 4 for an example) are unsustainable for the current and future testing needs in medical education.

Purpose of Study

I will develop, demonstrate, and document a systematic process for generating test items using an item modeling methodology. Hence, the purpose of my dissertation is to introduce a framework to guide item generation and to present a proof-of-concept application demonstrating how items can be generated using this framework. My unique contribution to the literature is threefold. First, I will present a modeling approach for extracting knowledge from content experts that can be used for item generation. Second, I will present an item generation technique that can be used to minimize text similarity between generated items from the same model thereby ensuring the new items are diverse in content. Third, I will present a method for evaluating text similarity for the generated items. These methods and procedures will be demonstrated in the context of medical education, more specifically in the content area of surgery.

In short, educational test items are in severe demand. The transition from paper to computer-based testing is dependent on the adequate supply (i.e., large numbers) of test items. Research on innovative test designs also relies on the availability of test items. AIG serves as one possible method to overcome the costly and time-consuming process that currently exists for creating test items. AIG has the potential to minimize item development costs, eliminate item exposure concerns, promote effective test development, enable sophisticated and informative test designs to be implemented in operational testing programs,

and, most importantly, provide test items in a volume, quality, and manner that test developers now demand.

Organization of the Dissertation

My dissertation is organized into six chapters. The current chapter, Chapter 1, is an introduction to my research area as well as an overview and justification for this study. Chapter 2 is a literature review of past attempts and current perspectives on item generation. Chapter 3 is a description of my proposed framework for modeling and generating test items. This chapter includes a description of the item generation process required to transfer content expert knowledge into a modeling framework for item generation. Chapter 4 contains the application of the generative framework, where I demonstrate how the framework can be applied to generate test items in the medical content area of surgery. In Chapter 5, I present a method to evaluate the quality of an item model by measuring item similarity after generation and providing a demonstration using items generated from the previous chapter. Chapter 6 contains a summary of this study, a description of the limitations of the study, and future directions on how the item generation framework can be realized in educational assessment.

Chapter II: Literature Review

In this chapter, I present a summary of the item generation literature in four sections. First, I present a short history of the ideas and developments that led to what is now commonly referred to as automatic item generation (AIG) in the educational measurement literature. Second, I review developments associated with how information must be organized to generate test items. More specifically, I focus on the current approaches to item modeling. Third, I review current developments in the technology-based processes required to actually generate the items. Fourth, I present developments required after items have been generated, meaning the use of statistical methods to estimate the psychometric characteristics of the generated items. After this four part review of the literature, I identify the limitations of our current AIG practices. These limitations will be addressed in my dissertation research.

A Concise History of AIG

The demand for large numbers of new test items began when developers (e.g., assessment practitioners, content specialists; item writers) of traditional paper-based tests began to address test security and test fairness issues (Roid & Haladyna, 1982). Security concerns originally emerged when tests were administered on a single form. The use of one form meant the content could easily become exposed because all examinees viewed the same set of items. To address this potential security problem, parallel test forms were developed. Parallel forms are multiple tests containing different items that measure the

same content and construct but are designed to be administered concurrently to minimize item exposure. Because parallel forms are used interchangeably, items on each form must be equivalent in their content and statistical (e.g., same item difficulty level) specifications. Parallel form development requires large numbers of items with comparable content and statistical characteristics. The demand for items to assemble parallel test form led to a predecessor of AIG called systematic item writing.

Early Developments in Writing Educational Test Items

In the process of writing items for parallel forms construction, an important limitation of the item development task itself was revealed: subjectivity. When an item is written, it represents an expression of the content expert's own concept of the knowledge, skills, and competencies within a particular content area. This expression, unique to each expert, introduces subjectivity into the item development process and renders each item unique. Hence, item writing quickly became conceptualized as an "art" which relies on the knowledge, experience, and insight of the particular item author. Parallel test forms construction, however, requires that items on different forms have comparable content and psychometric characteristics. This need to produce fair tests using a subjective item development process prompted researchers to introduce item writing guidelines in an attempt to control for the potentially diverse outcomes that could be produced when item writers were asked to perform an identical task (Ebel,1951). Guidelines provide a summary of best

practices, common mistakes, and general expectations of item writing to ensure content experts have a shared understanding of the task at-hand (Haladyna, 1994; Case & Swanson, 1999). Today, item writing guidelines are widely adopted in testing programs and popular among item writers (Haladyna, 2002). Despite the use of these guidelines, the issue of subjectivity remains an inherent feature in the item writing process.

Another way to minimize item writing subjectivity is systematize the process. Systematic item writing was first proposed by Guttman in 1957 when he suggested that multiple test items can be created from the same facet of knowledge. Guttman's approach relied on facet theory, which is a systematic approach for organizing related knowledge into a common component known as a "facet" (Guttman, 1957). Guttman's approach led to the creation of new items using a method of knowledge substitution. For example, if different types of acid are conceptualized under a common facet of knowledge, then multiple items can be created when different acid types (e.g., hydrochloric, bromic, sulfuric) are substituted into the same item structure. Guttman's approach highlighted how subjectivity in item writing can be minimized by restricting the item writing process so that a more standardized and, hence, objective test item would be created.

Building on the logic of facet theory, Bormuth (1970) proposed a rule-based approach to item writing. Again, with the goal of minimizing subjectivity

and maximizing objectivity during the item writing process, Bormuth used prose transformations and algorithmic expressions to create items from prose that contained the knowledge to be assessed (see Figure 1). With this approach, verbs and nouns are replaced to produce semantically diverse items. Although Bormuth's approach was recognized as a good solution for writing text-based items by some of his colleagues (e.g., Follman, 1971), the logic behind transformational grammar was little known to the test development community at the time. Bormuth's approach was also limited to text-based items. Hence, with a lack of understanding and no broad utility in educational testing, at least, at the time few researchers or practitioners adopted or advocated for Bormuth's approach to item development (Haladyna, 1995).

Prose:

1. **Chormium oxide** is used as **a polishing agent** for **stainless steel**.
2. **Tin oxide** is **a pure white powder** used extensively as **a polishing agent** in **dentistry**.

Structure:

[Key Noun] (Descriptor) [Application] [Context of Application]

Item Structure:

What is used as [Application] for/in [Context] ?

Items:

1. What is used as a polishing agent for stainless steel?
2. What is used as a polishing agent in dentistry?

Figure 1. An example of Bormuth's (1970) approach to item writing using transformational grammar, adapted from Roid and Haladyna (1982).

The first simple framework that produced a systematic and generalized method to item writing was called *item shell* development (Roid & Haladyna, 1982; Millman & Westman, 1989). A shell contains an outline of the item's syntactic structure. This structure is then replicated to produce new items by manually substituting content into the shell to produce similar items. Unlike Bormuth's approach, item shells were concrete, more generalizable (e.g., could be used with text and numeric content), and hence more understandable to practitioners (Haladyna, 1995). Test developers, for the most part, embraced

the use of item shells because it was clear how the structure could be manipulated to create similar items (see Figure 2). Following Guttman's (1957) logic for item writing, an item shell requires the substitution of content into variable slots so new items could be created. Although shells yield items that look very similar to one another using a labour-intensive and manual substitution process, it serves as the first systematic approach to item development that can be considered a success among content specialists (Haladyna, 1995) to the task of generating test items.

Parent Item:

What is the distinguishing characteristic of hydrogen?

Item Shell:

What is the distinguishing characteristic of (gases studied in this unit)?

Items:

What is the distinguishing characteristic of Nitrogen?

What is the distinguishing characteristic of Oxygen?

What is the distinguishing characteristic of Hydrogen?

Figure 2. An example of an item shell adapted from Haladyna (1994).

Up to this point in the historical developments, item generation methods are focused on creating items on a one-by-one manner, where the mechanism of

generation automates the process of creating individual items. With the introduction of *item modeling* (LaDuca, Staple,s Templton, & Holzman, 1986; Bejar et al., 2002), the unit of analysis change from the individual items during generation, to groups or sets of generated items. Hence, item modeling serves as the first modern approach to automatic item generation. Current item generation efforts can be categorized into three different areas relative to the temporal process used for item generation. Before item generation, the focus is on item model development. During item generation, the focus is on technology-based solutions for the actual production and assembly of items. After item generation, the focus is on statistically modeling the psychometric characteristics of the generated items so these items can be banked and used for test assembly (e.g., automated test assembly) or test administration (e.g., computer adaptive testing). In the next three sections, I provide an overview of each aspect of item generation and describe the approaches currently available in the research literature.

Before Item Generation: Item Modeling

The distinctions between an item shell and model are subtle but important. The purpose of both the shell and model is identical; that is, they are both structures used to create new items. Shells, however, differ from models in how they are conceptualized. In the past, existing test items were modified to form item shells (Haladyna, 1995). Specifically, shells were created by manually substituting different but related pieces of information into a variable slot to

create items, one-at-a-time. This approach ensures that the generated items have comparable statistical properties because the content used for the substitution process can be carefully selected. This approach to item generation is also called cloning (Glas, 2006). Conversely, item modeling is an inductive approach for generating items where structures of knowledge can guide the generation process.

Item Model

An item model is a knowledge structure used to generate test items. It differs from an item shell by having a modular structure composed of elements and components. First, there are parts of item models that may vary from one generated item to another. This variable part is known as an *element*. Second, different parts of an item model can be used to present different information within an item. These sections, called *components*, also vary in how they are defined. These key parts of the item model, elements and components, are described in more detail.

Elements

To operationalize an item model, it must contain at least one element. An element can take on many different forms. For example, an element can be a part of a sentence, a whole sentence, a key term, or a numerical value. Most importantly, elements are parts of an item model that include variable content. This description of an element was first presented by Bejar et al. (2002). In

Figure 3, element *S1* describes different occupations. In this case, the element is a string or non-numeric value. Elements may also be numeric or integer values. This differentiation was needed because strings and integers are often treated differently within an item model. String elements, for example, are usually denoted as a whole unit of information (e.g., text) whereas integers require calculations and ranges, often for quantitative manipulations. The variable *S1* in Figure 3 is a string variable containing text content (i.e., occupations). We also define how *S1* should vary between each instance of a generated item when we create the item model. With numerical elements such as *I1*, *I2* and *I3*, they vary within a specified range and this step range is manipulated systematically during item generation. In short, this item model highlights the variations between integer and string elements.

Item Model Components

Stem	Gina works as a S1. She works I1 hours per week and makes $I2 per hour. She is paid weekly and I3% of her weekly wage is deducted for taxes. If Gina saves 1/4 of her paycheck every week, then how much money will she have saved at the end of 4 weeks?
Elements	S1 Range: “dishwasher”, “cashier”, “landscaper”, “receptionist”, “telemarketer” I1 Value Range: 25 – 40 by 0.5 I2 Value Range: 9 – 12 by 0.25 I3 Value Range: 5 – 9 by 1
Options	A. I3/100*I2*I1 B. (I3/100) *I2*I1 C. (1-(I3/100))*I2*I1*4 D. (I3/100)*I2*I1*4
Key	C

Figure 3. Example of an item model in mathematics.

Radicals

Bejar (2002) claimed item generation can yield two types of items: variants or isomorphs. Models that generate variants produce items that are expected to be psychometrically different from one another (e.g., items that

have different difficulty levels). Since generated items only differ by their varied elements defined within the item model, elements that affect the psychometric properties are known as *radicals*. That is, radicals directly affect the psychometric characteristics of items. To ensure generated items are psychometrically comparable, LaDuca et al. (1986) claimed that item models should not contain radical elements to ensure the psychometric quality of the generated items are comparable (i.e., generated items are clones). But with a wealth of psychometric information, radicals can now be used to generate items across a range of difficulty levels (Bejar, 2002). An example of a radical can be found in Figure 3, where element *I3* varies from 15% to 25%. If information on the examinee's performance level reveals that a particular skill, such as solving item with manipulations of 20% is easier than solving item with manipulations of odd percentages, then *I3* would be a radical element.

Incidentals

Incidentals are elements that perform in the opposite manner to radicals, meaning the variation of incidental elements should yield psychometrically similar or even identical generated items. If an item model only contains incidentals, then the generated items from that model are known as isomorphs because the generated items are expected to have similar psychometric characteristics (Gierl, Zhou, & Alves, 2008). An example of an incidental element is shown in *S1* of Figure 3, where variations of the *S1* element should not change the psychometric properties of the generated items, as each

variation would only change the context but not the complexity, hence, difficulty level, of the item.

To further Bejar's (2002) claims, Embretson (2002) stated that two additional factors would influence how elements affect the psychometric characteristics of the generated items: the number elements used and differences between variations of each element. For example, a smaller number of elements in a model should yield more similar items. Conversely, if a large number of elements are manipulated, then the generated items should be more different from one another. Generated items that are more diverse from one another are also expected to have more variable psychometric characteristics. Embretson also argued that element variation may produce different levels of cognitive complexity which, again, may affect the psychometric characteristics of the generated items. The issue of how elements are manipulated and the effect of these manipulations on the psychometric characteristics of the generated items will be discussed further in the statistical calibration section of my literature review.

Components

In addition to elements, item models are organized into different components for expressing information in a generated item. The components used in an item model depend on the item type. The most common item format

we currently use, selected response or multiple-choice items, typically contain two components: stem and options.

Stem

The stem is a component of the item model where the question or prompt is posed to the examinee. The stem contains information that allows the examinee to understand the prompt. Gierl, Zhou, and Alves (2008) claimed the stem could be manipulated in four different ways to influence item generation. First, elements in a stem may be independent from one another, meaning variation of any elements in the item stem is not related to the variation of another element. Second, elements may be deployed dependently, meaning variations in one element affects variation of other elements. Third, the elements in the stem may be varied both dependently and independently in a mixed format. This type of stem also implies that the model contains at least three elements. Fourth, a stem is fixed, meaning no elements are used in the stem. This implies the same stem would be used for all items generated from the model.

Options

Options are a set of alternative responses examinees must select when answering the test item. Along with the correct response known as the key, a set of plausible responses are presented to serve as distracters or incorrect options. Since the generated item stem may change the keyed response for the

item, options in item generation often will need to be adjusted accordingly. For example, the options in Figure 3 are defined such that the key and distracters are calculated based on each generated item stem. Gierl et al. (2008) described three types of options: randomly assigned, constrained, and constant. With randomly assigned options, distracters are chosen at random from a list of options. Such options are ideal for generating items that do not require calculation. Constrained options are limited by the elements of a generated item. These options are often used when the solution for the generated item requires some type of calculation. Constant options are fixed options presented in the same way for all generated items.

In sum, an item model is composed of different components, and each component contains a different number of elements. To organize the possible variations in item models, Gierl et al. (2008) suggested a taxonomy of item model types based on an interaction of different stem and option characteristics. Figure 4 provides an illustration of the taxonomy. The shaded boxes represent infeasible combinations because they only generate one item.

Options	Stem			
	Independent	Dependent	Mixed	Fixed
Randomly Selected				
Constrained				
Fixed				

Figure 4. Illustration of the Item Model Taxonomy described in Gierl et al. (2008).

Current Approaches to Item Model Development

Two theoretical approaches are used to guide item model development. These approaches are called strong and weak theory (Bejar, 2002; Drasgow, Luecht, & Bennett, 2006). These theories help conceptualize how content is identified and combined for the purpose of generating test items.

Strong Theory

Ideally, items are generated with precision such that they are designed to measure specific sets of skills and knowledge. To do so, item generation requires a comprehensive knowledge structure that can help guide how the content within elements are varied and combined. Therefore, in addition to

producing large numbers of items, it is also desirable to have an item generation process that can ensure the content in the items represent the knowledge structure and processing skills used by examinees to solve test items. Strong theory refers to this deterministic approach to generating test items.

To enable strong theory AIG, cognitive models offer an organization of the knowledge and skills that can be used to order complexity, typically operationalized as the difficulty level of the generated items (Drasgow et al., 2006). Therefore, strong theory refers to the use of any cognitive model or knowledge structure that provides an estimate of the psychometric property for each of the manipulated components in a set of generated items. Drasgow et al. claimed that by modeling the interaction between the examinee and the content, it is possible to control and therefore predict the psychometric characteristics of the generated items. This approach, when successful, has the added benefit of yielding a strong inferential link between the examinees' test item performance in a specific content area with the interpretation of the examinees' test score because the model features that elicit the item characteristics are predictive of test performance (Bejar, 2012).

Cognitive models are beginning to play an important role in educational measurement (Leighton & Gierl, 2010). Applications of strong theory AIG generated test items in the psychological testing domain (Embretson, 2002), where robust cognitive models exist. For example, strong theory AIG has been used with spatial and analogical reasoning tasks to generate items (Bejar, 1990;

Janssen, 2010). An example of strong theory AIG used in the domain of K-12 testing was described by Gierl and Lai (2011), where items were generated in the domain of high school algebra. Figure 5 illustrates how strong theory can be used to develop item models to generate items that measure factoring multiple variables in algebraic expressions.

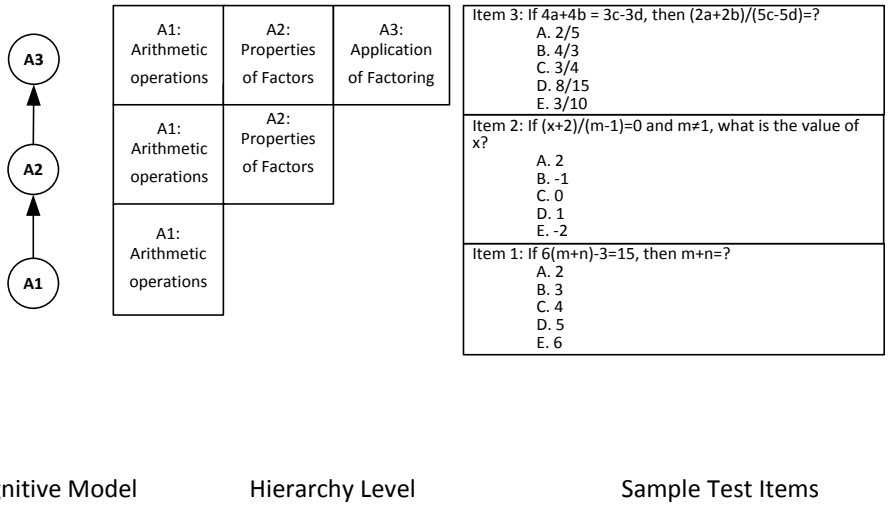

Figure 5. Diagram of a general description of developing item models with strong theory from Gierl & Lai (2011).

As more test developers begin to realize the importance of cognitive modeling in test development, educational testing is beginning to apply strong theory to generate items in domains where few established models currently exist. For example, to demonstrate the use of a strong theory approach for generating items in medical education, Gierl, Lai, and Turner (2012) introduced the concept of a cognitive modeling for AIG in the area of medical licensure

testing. Despite a small number of demonstrations, strong theory has not been widely adopted in the test development community. The problem of using strong theory AIG rests, in part, from the fact that we lack well-defined structures of knowledge for creating test items in some educational domains (Leighton & Gierl, 2010). Also, the fact that we currently lack a methodology for strong theory AIG also helps account for its limited use. In sum, strong theory item modeling is theoretically sound, but the required knowledge and resources are not yet available to researchers and practitioners to permit this approach to become widely applicable.

Weak Theory

In lieu of a well-defined cognitive model or knowledge structure for generating items, guidelines from best practices are typically used to direct item generation. These guidelines, developed largely from the experience of test developers, form the basis of a weak theory approach to AIG. Recall that radicals are elements in an item model that affect the psychometric characteristics of the generated items. In weak theory, radicals are developed at the discretion of test developers who identify features from existing items that are expected to affect difficulty. An example of how weak theory AIG can be used for item model development is presented in Figure 6. In this example, items are developed to probe Grade 3 examinees' understanding in skip counting. Integer *I1* in Figure 6 can be any number. If this number was with strong theory, then there must be a cognitive model or knowledge structure to specify how counting backwards from

a different range of numbers require a different set of skills thereby altering the difficulty of the item. In the absence of such explanatory knowledge, test developer and content specialists *anticipate* how students counting from multiples of 5s will solve this item. Therefore, *I1* and *I2* are specified as integer values that are expected to yield items with similar psychometric properties. This expectation is not as theoretically defensible as it would be if an articulate strong theory approach was used. Rather, this expectation about the psychometric characteristics of the generated items is based on the experience and judgement of the content specialists.

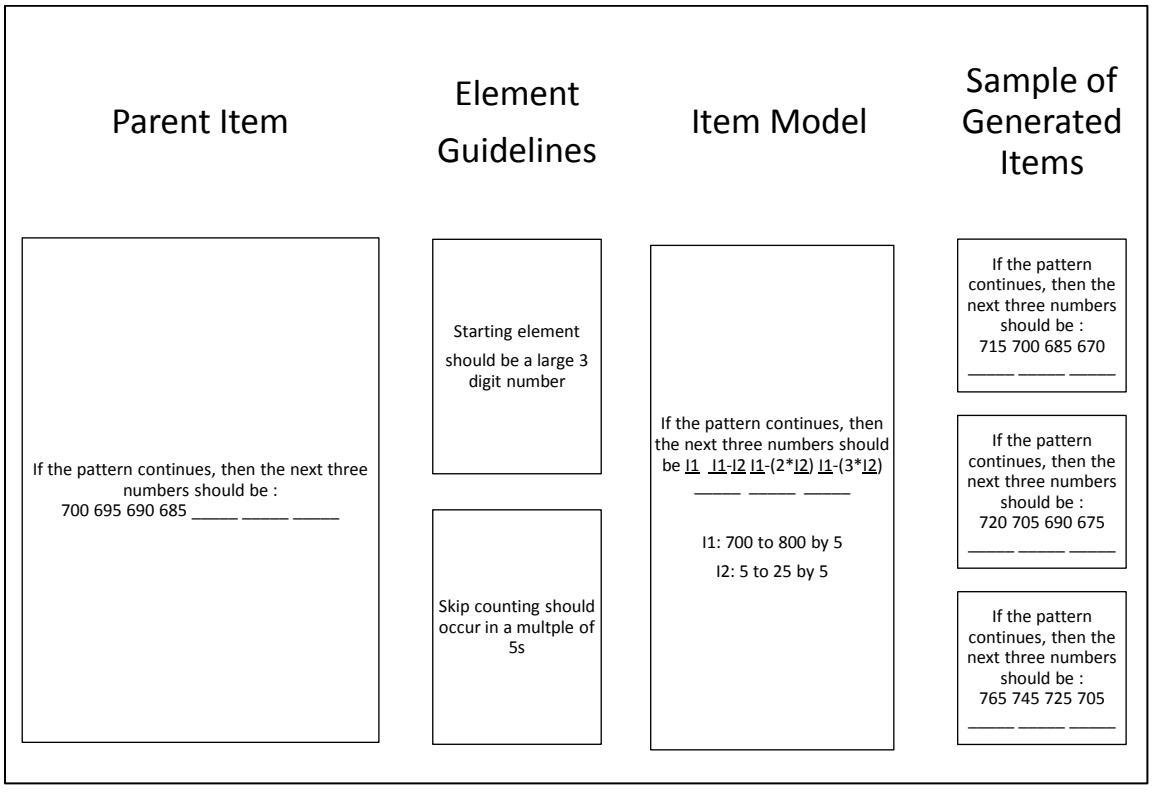

Figure 6. Illustration of developing item models using weak theory.

An item model developed from weak theory is comparable to the item shell approach. However, one key difference resides in how the elements are varied. With an item shell, elements are created to address the purposes of test, meaning that item writers can insert any value in the element if it produces items that are suitable for the current testing situation. Alternatively, with an item model, the developer is required to provide a range of all possible alternatives that are expected to yield generated items with similar psychometric characteristics.

Admittedly, it is less desirable for items to be generated from weak theory, as there is less evidence to justify the manipulation of elements for item generation. There is also less evidence to draw on for predicting the psychometric characteristics of the generated items. But, weak theory AIG is also a practical solution for item generation given that articulate cognitive models and knowledge structures typically do not exist to account for problem solving in many content domains. The practicality of weak theory item generation was demonstrated in Lai, Alves, and Gierl (2009), where a panel of eight item writers were trained to create their own item models. These models, in turn, were used to generate 64,280 items. Items were also generated for a wide range of grade levels and across all curriculum-specific content areas in science, mathematics, and language arts. This outcome is in contrast to generating items from strong theory AIG, where the generated items have a much narrower scope in content and skills. For example, Lai, Gierl, and Alves

(2010) generated 10,278 items in Mathematics using strong theory, but the generated items were only designed to probe two areas in mathematics (applied word problems and exponential functions) under one topic (understanding algebraic solution).

Summary of Item Modeling Approaches

In this section, I presented aspects of item modeling, the current approach to organizing information prior to item generation. I described how item models differ from previous approaches to item generation and I operationally defined the different components in an item model. Then, I presented the two frameworks that can guide how item models are created. Baranowski (2006) likened the development of items as an "art" form, which is a sentiment shared by many researchers and practitioners in educational testing. In contrast, there is now growing support that suggests item writing should be a science (Bormuth, 1970; Hornke, 2002; Haladyna & Gierl, 2012). The two approaches to item generation, strong and weak theory, parallels this ongoing struggle to determine whether item generation, an item writing approach designed to produce a larger pool of items, is an art or science. In the next section, I review the technological aspects of item generation and describe how items can be generated from the models described in the current section.

During Item Generation: Technology and Content Assembly

Producing multiple items from a shell did not require a generative mechanism because test developers simply substituted different values into the task using a manual process to produce new items. But the demand for concurrent item generation led to technological innovations where content is combined within an item model using an automated process. In this section, I first present an overview on the concepts that guide the technology necessary for item generation. Then, I provide a review of the item generation software that is currently available. Finally, I highlight the challenges that must be addressed to improve on the current technology used for generating test items.

Key Concepts in Item Generation

The development of technological solutions to address item generation problems can be traced to fields outside of educational measurement, where each attempted solution was created, in part, to meet the challenges in their own respective fields. In general, existing item generation solutions can be categorized into two distinctive types: artificial intelligence-based (AI-AIG) and template-based (T-AIG) AIG. AI-AIG focuses on automating the entire traditional approach to item development. Technology is used to replace item writers in order to gather, summarize, and generate test items from electronic sources of information. The automated processes are designed to computationally mimic the tasks performed by an item writer. Alternatively, T-AIG focuses on developing a hybrid approach to item generation, where content experts provide

information for the item models and technology is used to generate items from each model. Both approaches to item generation are summarized next.

Artificial Intelligence-Based Automatic Item Generation (AI-AIG)

In computer science, it is anticipated, by some, that item generation tasks will dominate research in the field of natural language processing for the next decade (e.g., Graesser, Rus, Cai, & Hu, 2012). To enable this endeavor, computer scientists use recent advances in the field of natural language generation, information processing, and artificial intelligence (van den Bosch & Bouma, 2011). Advances in test item generation from computer science do not require the use of a template, as templates are not deemed to be a true approach for natural language generation (Deemter, Krahmer, & Theune, 2005). One advantage of generating items without a template or item model is flexibility in item generation. As a result, computer scientists have offered a diverse set of solutions in the past decade to the challenge of item generation.

A review on the state-of-the-art in item generation techniques was recently presented in Graesser et al. (2012). A common theme in all AI-AIG approaches is that they require similar efforts to question answering. For example, when a computer is tasked with answering a question, AI programming is used to evaluate the content within the question, where both context and content must be assessed, and the list of likely responses is determined. The list of likely responses are then reworded and analyzed based on the original

content and context to ensure the computer will respond in a coherent manner (Bosma, Marsi, Krahmer & Theune, 2011). In AI-AIG, different sources of information are first processed using AI programming to search for likely content that can be used to generate a test item. Related content and context are determined and used, along with human created sample questions, to formulate an item that is deemed to be appropriate.

An example of how items can be generated directly from a corpus of text was demonstrated by Karamanis, Ha, and Mitkov (2006). They presented a five-step generation process to extract test items directly from textbooks. First, a relevant text corpus is parsed into different units of analysis (e.g., sentences, segments of words). Second, key terms appropriate to the domain are identified by content experts. These key terms are highlighted throughout the corpus. Third, clauses within the corpus are categorized into different semantic knowledge types for use with potential questions. Fourth, clauses of different types are transformed into item stems based on their semantic knowledge type. Fifth, distractors are selected from the correct answer based on appropriate levels of semantic distances. Liu (2009), more recently, expanded on this approach by structuring the process as a modular framework of AI-AIG that can be used to generate items from a variety of electronic resources.

Although AI-AIG approaches are technologically impressive, research often focuses on the techniques for how items can be generated, and rarely

describe the generated items as educational assessment tasks. For instance, Karamanis et al. (2006) generated 279 items from a corpus of 32,043 words in the medical education domain. But, after a review of the generated items, content experts determined that only 19% of the items were deemed useful for testing. Karamanis et al. demonstrated that natural language processing (NLP) methods have the ability to synthesize test items from a passage of text, but it is also evident that a review process is needed to ensure the generated items are meaningful for testing. To minimize item reviews and to increase the number of usable items generated from this process, Foster (2010) suggested the use of statistical NLP methods to measure semantic and syntactic similarities between two text passages to refine the corpus and reduce the number of unusable items. Foster's approach is comparable to the method used by Lai and Becker (2010) to identify items that measure similar content and therefore should not be administered together (i.e., enemy items) using statistical NLP methods.

Because current research on AI-AIG is focused on the generation process for commercialization, most software solutions are proprietary, meaning no comprehensive documentation or descriptions are available. Two proprietary examples are presented to illustrate some aspects of an AI-AIG solution. Gutl, Lankmayr, Weinhofer, and Hofler(2011) developed software called the *Enhance Automatic Question Creator* (EAQC) to generate test items under the AI-AIG framework. Following other AI-AIG approaches, they outlined a prototype software that computes the AIG task in three steps. First, information from

different sources (e.g., Internet, written documents, etc.) are processed into formats that can be used for item generation. In this first section, the document is broken into segments of texts in different lengths where counts of occurrences, length of sentences, and relation of words within segments are calculated. Second, different segments of the document are organized into concepts. The concepts are separated using statistical analyses of the structural and semantic information of the text calculated in the first section. Information is also calculated for weighting based on different sources (e.g., relation to other sections of text, relevance of words in each section of the text). Third, using this information from each concept, items are generated based on the appropriateness to the item format. Gutl et al. (2011) presented a pilot study using one essay as the information source to produce 16 generated test items. Although there were errors in the generated items (e.g., one item was "What do you know about Natural Language processing in the context of Natural language processing?"), the mechanism demonstrated the potential of the AI-AIG approach for educational testing.

The second software example for AI-AIG was developed at ETS. It is called the *Item Distiller* (Higgins, 2007). Rather than automating the entire process of item generation, the *Item Distiller* uses AI to gather and search information through a corpus of information to identify patterns for constructing item models. To enable this process, users enter a search criterion such as a specific sentence pattern. Then, the *Item Distiller* searches through different patterns of

information to formulate an item model that can be used to generate new items. This software demonstrates how AI-AIG can be implemented systematically without requiring the content specialists to initially create the item models.

To summarize, exciting and innovative research is now taking place on AI-AIG. Clearly, some hurdles remain before AI-AIG can be realized. Excluding the obvious challenges of computational complexity and high development costs, AI-generated items can be produced. However, these items still require a traditional review process from content experts to determine their suitability for assessment. With item requirements ranging in thousands or tens of thousands, the current approaches to AI-AIG would not reduce test development workload, but merely shift it from creating to reviewing items.

Template-Based Automatic Item Generation

In natural language processing, template-based solutions have long been used as an alternative approach for language generation. Template-based technology requires a human to structure prototypes that are used, in turn, to propagate the language generation processes (Reiter & Dale, 1997). Deemter, Krahmer, and Theune (2005) argued that, depending on the specificity of the required task, template-based approaches can be just as effectively as true AI approaches. With a specific task such as generating test items, template-based automatic item generation (T-AIG) is an approach that has been described in educational measurement (Deane and Sheehan, 2003; Drasgow, Luecht &

Bennett, 2006). Compared with AI-AIG, T-AIG is more constrained, where templates in the form of item models, are created to fulfill a specific set of requirements from the content domain. Different mechanisms are then used to generate items. For instance, Hornke (2002) summarized how items can be generated for mental rotation and pattern matrices. Embretson (2002, 2010) demonstrated how items can be generated to probe object assembly and abstract reasoning skills. Arendasy & Sommer (2010) generated items from continuous 3D images to assess spatial-rotation skills. Bejar et al. (2002) developed a template-based AIG method for on-the fly testing in mathematics.

T-AIG relies on the use of models for generating items. A well-defined item model should minimize the psychometric variations and maximize the semantic differences between tasks generated from a single model (Deane & Sheehan, 2003). The use of item modeling also permits technologically simple and efficient approach to item generation. Next, I describe a sample of T-AIG software programs that are currently available to generate items.

Compared to AI-AIG, more software have been developed to generate items using templates. Although no program is commercially available yet, more choices are available for use compared with AI-AIG. With a well-defined item model, generation can be a relatively straight-forward process (Lai, Alves, Zhou, & Gierl, 2009). Generating items from a template-based approach is an exercise

in iteration, meaning computer programming can be used to instantiate every unique combination of elements in an item model, subject to constraints.

Following this logic, ETS developed software to complete the item generation task. Singley and Bennett (2002) used the *Math Test Creation Assistant* (MTCA) to generate items involving linear systems of equations. By systematically manipulating numerical units and varying a range of variables (e.g., distance, temperature), the MTCA combined different elements to generate items in mathematics. This template-based approach was later generalized to produce items in different languages using a program created by Higgins, Futagi, and Deane (2005) called *ModelCreator. ModelCreator* requires a database of translated components used for item assembly, where templates for different languages are assembled using varying sentence structures. Gierl et al. (2008) presented another T-AIG solution called *Item Generator* or IGOR, for short, to generate multiple-choice and constructed-response test items. The software was developed using the JAVA programming language so it can be executed under different computing platforms. All input and output files from IGOR are implemented in Extended Mark-up Language (XML), meaning data for each item model or generated items are stored in a hierarchically tagged text file (see Figure 7). The use of this file type permits both the models and the generated items to be seamlessly transferred to different software (e.g., databases, word processors, item banking or spreadsheet applications). To generate items using IGOR, an item model must be expressed in this

programmable format. Figure 8 is an illustration of the graphical user interface (GUI) that can be used to program item models into IGOR.

```
<Template>
        <Stem>
        <Value>I have [[I1]] tens, [[I2]] hundreds, [[I3]] ones.  What number am I?</Value>
        </Stem>

        <Variable name="I1" type="Number">
        <Range max="17.0" min="11.0" step="2.0"/></Variable>
        <Variable name="I2" type="Number">
        <Range max="5.0" min="2.0" step="1.0"/>   </Variable>
        <Variable name="I3" type="Number">
        <Range max="29.0" min="15.0" step="2.0"/>              </Variable>

        <Option group="Key" type="Equation">
        <Value>([[I1]]*10)+([[I2]]*100)+[[I3]]</Value></Option>
        <Option group="Distracter" type="Equation">
        <Value>([[I2]]*100)+([[I1]]+[[I3]])</Value></Option>
        <Option group="Distracter" type="Equation">
        <Value>([[I1]]-10)+[[I3]]+(([[I2]]+1)*100)</Value></Option>
        <Option group="Distracter" type="Equation">
        <Value>([[I2]]*100)+[[I1]]</Value></Option>
</Template>
```

Figure 7. An item model formatted in XML.

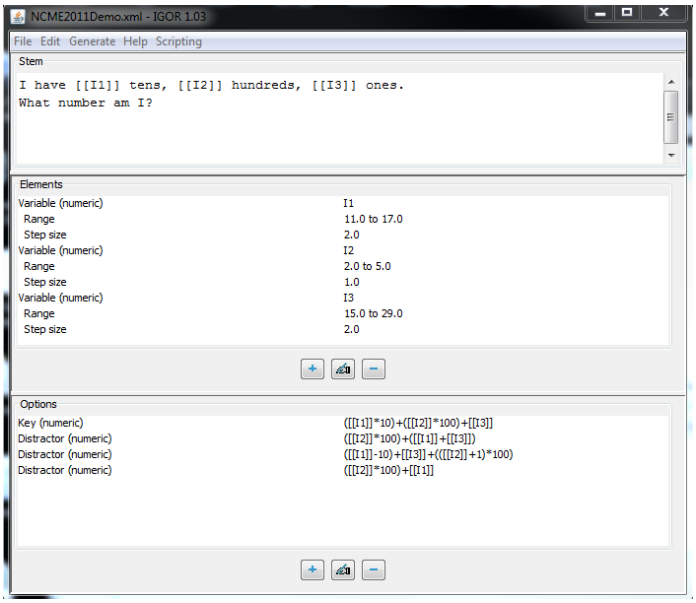

Figure 8. Graphical User Interface of IGOR.

The use of item models in T-AIG permits a simple generation process using an iterative approach. The generation process can be applied to produce diverse items. As the generation process iterates through all desirable combinations of elements in an item model, control over the quality of the generated items is dependent on the item model. As a result, item quality is established through the scrutiny of content specialists and through the design considerations of the item models. But the use of T-AIG methods for generation presents two limitations, generation of syntactically similar items and content area limitations.

The use of T-AIG often leads to the criticism that generated items are too syntactically similar (Bejar et al., 2003). To minimize item exposure and security risks, generated items from the same item model need to be semantically similar

but syntactically different. If generated items are too syntactically similar to one another, then administering such items not only risks exposing the item model, but it yields large numbers of items that may not be usable in operational testing. One solution to avoid the generation of overly similar items is the use of incidental elements to mask the similarity between generated items (Bejar, 2002). But incidental elements still limit variation in items because the number of potential elements in any one item model is, typically, small. For example, if an item contains 15 words in the stem, then the maximum number of elements that can be manipulated is 15, assuming that each word can be made into an element. The current solution to overcome similarity in generated items is to develop more complex item models. But as item models become more sophisticated, additional constraints are required to produce feasible items. Alves, Gierl, and Lai (2010) developed item models for high school Biology that required five to seven separate programming constraints for each item model. Experienced test developers are aware of the rules and modifications needed to create comparable items (Haladyna, 1995). But to express these rules in an item model, test developers may not have the necessary programming knowledge to operationalize these constraints (Singley & Bennett, 2002). With no available guideline for developing item models (Gierl & Lai, 2011), ensuring quality and feasibility across the generated items becomes very challenging for test developers.

To-date, systematic item generation approaches in educational testing are limited to a few, somewhat predictable, content areas. While impressive number of items can be generated in areas such as mathematics, other areas such as language arts have had little or no success. Methods such as transformational grammar (Bormuth, 1970) or item design rules (Hornke, 2002) have been proposed for verbal analogies and reading comprehension item generation, but the number of items produced from these models are comparatively small and disappointing (Lai, Gierl, & Alves, 2010). One possible reason for the lack of success in generating text-rich content areas like language arts and social studies may be related to the generation mechanism itself. Often, few components in a language art item can vary systematically because the task is based on facts (also called "factoids" in the AIG literature). Factoids are highly constrained problems for the generation process as only a small amount of specific information can be manipulated for a single factoid. Therefore, item models generally yield fewer items with T-AIG when facts are used for item generation. Moreover, language-based items may be more dependent on the gathering and presenting of information than items from other content areas. For example, it is common to create multiple items from a single reading passage where different set of passage-based items can be administered on the same test. However, the number of variations that can be extracted from this fixed, fact-based, passage is often small. Therefore, the developments of item models have not been successful in all content areas and,

as a result, applying the current item generation solutions to different content domains is not uniformly successful.

After Item Generation: Models for Statistical Calibration

The third section of my temporal-based review of different item generation methods focus on the analyses that are conducted after test items are generated. The process of determining the psychometric property of the generated items is known as calibration. In traditional item development, calibration generally takes place after items are administered to a field or pilot sample of examinees. Field testing can be completed by either appending items to existing operational test forms or administering a separate testing process. Since item generation yields thousands of new items, traditional field testing methods are no longer feasible because large numbers of items would need to be administered. The final research area I summarize, therefore, focuses on how statistical information can be estimated for generated items without the need for extensive field testing. Drasgow et al. (2006) envisioned an ideal item generation process, where items are generated, calibrated with psychometric procedures, and assembled for administration, all without human intervention. To accomplish this goal, Bejar (2010) highlighted the research required to enable this ideal process in his summary of three future AIG developments:

1. A theoretical basis is required to account for the variability among the psychometric parameters (e.g., item difficulty) of the generated items from item models.

2. A statistical method is needed to estimate the psychometric parameters for each generated item using key characteristics or features of the items. This requires attention to the statistical procedures for estimating the effects of those characteristics or features.

3. A mechanism is required for generating items from item models.

These research goals highlight the need for statistical calibration as part of the generation process because item difficulty is often used to assemble test forms for administration. With AIG, an alternative approach to item calibration is required. Depending on the type of psychometric information needed, items can be generated with their associated psychometric parameters thereby minimizing or even eliminating the need for field testing. Current item calibration approaches follow the theory of a random item (de Boeck, 2006), where an item is an instance of the theoretical representation of the assessment task. Two general approaches for calibrating generated items stem from this ideal: Item-family or skill-based approaches to AIG. In the next section, I highlight how each approach can be implemented to estimate calibrated item difficulty and I summarize the statistical methods that are used to complete this process.

Item Family Calibration

Item family calibration approaches posit that if a collection of items represent isomorphic instances of known parameters, then statistical approaches are available to estimate the psychometric properties for the similar set of items. The existence of an item family comes from the literature on variability in statistical item calibration. If we consider that a set of generated items is assumed to be identical, then one parameter can be used to estimate the characteristics of one item from the entire item family, given an expected level of variability.

Item family calibration assumes that the generated items within the family consist of all incidental elements, where all items are assumed to be psychometrically comparable to one another. To illustrate how relatedness within an item family can be conceptualized, Sinharay, Johnson, and Williamson (2003) outlined a spectrum of item family relatedness, describing three appropriate estimation approaches for different levels of relatedness. On one end, there is the identical siblings model, where all generated items are identical to one another within the family. To calibrate this family of items, only one estimate is needed to describe all items because all items are identical. On the other end, there is the unrelated siblings model. To calibrate this family of items, each item should be calibrated individually as each item is not predictable from other items of the family. The third approach, named related siblings model (RSM), is a compromise between these two extremes. If items are

generated from a set of incidental elements, meaning multiple items should represent the same concept as the original item, then items in the family are expected to have a similar level of difficulty. From the perspective of estimating each item in the family, if all items are varied incidentally, then the difficulty of each item can also be estimated with a level of expected error. The calibration process is illustrated in Figure 9.

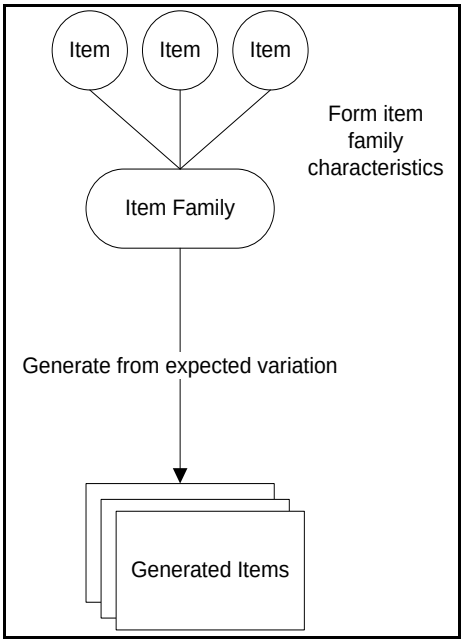

Figure 9. Illustration of the item family calibration process.

Using the related siblings family as the statistical model, items can be calibrated using Bayesian methods. Examinee responses to items from the same family are used to calibrate the parameters of the family, where an expected level of variability is estimated to account for the incidental variation of items for one family (Sinharay & Johnson, 2003). This Bayesian approach requires the

administration of items from the same families for estimating the psychometric properties of the item family.

Estimating the RSM requires three levels of information: item, item family, and person. Multiple items, each with a unique response function, can be represented as a subset of an item family. To estimate the probability of correct response for item *j*, traditional item response theory can be used:

$$P\left(X_j = 1 \middle| \theta, a_j, \beta_j, c_j\right) = c_j + (1 - c_j)\frac{1}{1+\exp(a_j(\theta-\beta_j))},$$

where the probability of a correct response is a logistic function of the given item's discrimination(a_j), difficulty (b_j) and its guessing parameter (c_j). Once item responses have been estimated, parameters for all items with the same family are transformed and extrapolated into a normal distribution,

$$\left(\alpha_j, \beta_j, \gamma_j\right)' | \lambda_{I(j)}, T_{I(j)} \sim N(\lambda_{I(j)}, T_{I(j)})$$

where for item family *I, λ is* the mean vector of the three parameters, *T* is the corresponding variance matrix for the three parameters, and the parameters are distributed under multivariate normal distribution. With item family variations calculated ($\lambda_{I(j)}, T_{I(j)}$) and assumed under a distribution, the probability of a correct response to any item *(j)* within family *I* can be calculated by

$$P(\theta | I(j)) =$$

$$\int_{\lambda_{I(j)}, \mathrm{T}_{I(j)}} \int_{\eta_j} P\left(\theta \middle| \eta_j\right) \phi_3\left(\eta_j \middle| \lambda_{I(j)}, \mathrm{T}_{I(j)}\right) d\eta_j f(\lambda_{I(j)}, \mathrm{T}_{I(j)} | X) d\lambda_{I(j)} dT_{I(j)},$$

where η_j is a vector of item characteristics for item j, ϕ_3 is the density function on the multivariate normal distribution for η_j, and $f(\lambda_{I(j)}, \mathrm{T}_{I(j)}|X)$ is the joint distribution of the means and variances of the family given the student response vectors.

The item family estimation approach was also implemented with hierarchical IRT modeling (Janssen, Tuerlinckx, Muelder & De Boeck, 2000; Glas & van der Linden, 2003). Glas and van der Linden(2006) demonstrated an application of this approach to calibrate generated items, and as its name implies, hierarchical IRT provide an item parameter through the estimation of the family parameters with an ordered level of specificity. Moreover, Shu, Burke, and Luecht (2010) demonstrated an application of calibrating generated items using simulated response sets. In sum, the item family approach of calibrating generated items is driven by the assumption that an expected level of variability can be estimated to predict the characteristic of generated items from the same family.

Skill-Based Calibration

An alternative approach to item family calibration is skill-based calibration. With a skills-based method, items are expressed as a set of required skills. Using strong theory AIG, a cognitive model or knowledge structure is composed of the knowledge, skills, processes, and competencies required by examinees to solve test items. These prescriptive knowledge and skills serve as

the radical elements that are varied systematically to promote item generation. These cognitive features are also expected to affect the psychometric properties of the items. Figure 10 presents a skill-based calibration process.

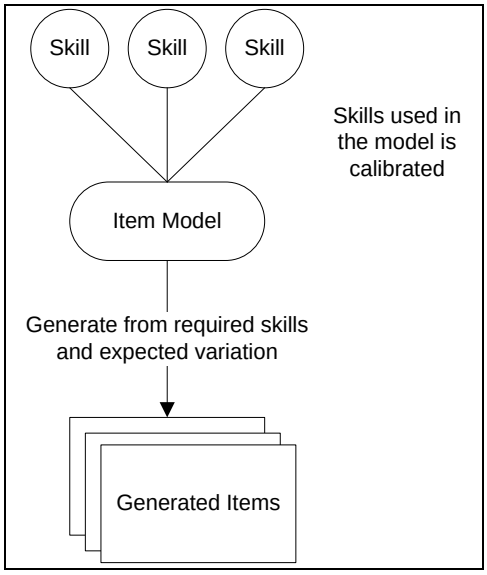

Figure 10. Illustration of the skill-based calibrated item generation.

Compared to item family approaches, skill-based calibration provides an explanation for how each required skill influences item difficulty. The earliest approach for skills-based calibration was presented by Fischer in 1973. It was called the linear logistic test model (LLTM; see also Embretson & Daniel, 2008). Instead of calibrating characteristics of each item using the examinee item response patterns, a logistic function is used to fit the likelihood of a correct response from a linear combination of required skills. In LLTM, multiple items can represent a set of required skills, where the effects of each skill on an item's difficulty is calibrated independently and each item is an instance of a set of

required skills. Using this statistical model, the probability of correct item response for item *i and* person *j* can be expressed as

$$P\left(X_{ij}=1\middle|\theta_j,q_{ik},\eta_k\right)=\frac{\exp(\theta_j-\sum_{k=1}^{K}q_{ik}\eta_k)}{1+\exp(\theta_j-\sum_{k=1}^{K}q_{ik}\eta_k)},$$

where *K* is the collection of skills required in the assessment, q_{ik} is a representation of the skill requirement on a given item, and η_k is the difficulty associated with skill *k*. This approach implies that difficulty of an item can be determined by

$$\beta_i=\sum_{k=1}^{K}q_{ik}\eta_k,$$

where items with the same set of required skills have the same difficulty parameter. LLTM also implies that item difficulty can be perfectly predicted by the required skills (Janssen, Schepers & Peres, 2004).

Embretson (2010) demonstrated a large proportion of the variance in any item response can be accounted for by two outcomes: the required skills needed to solve each item and the variation among items requiring the same set of skills. One model that can be used for skill-based calibration is an extension of the LLTM, proposed by Janssen et al. (2004), called the LLTM-RE, where RE are random effects. By estimating variability between items from the same set of skills, LLTM can be extended to include a random item effect. An illustration of the modification is presented in Figure 11. This diagram illustrates how the two LLTM models differ from one another. For the LLTM, the probability of an

examinee's correct response is estimated with an item difficulty parameter (B_i) and a person ability parameter (Θ_j). The person parameter is estimated from the true ability of the examinee (μ_j) and the estimated error on the ability of the examinee (ε_j), while item parameters are estimated from the use of prerequisite skills (q_{in}). The LLTM-RE, by comparison, estimates the parameter for items requiring the same set of cognitive features with a level of error in estimation thereby warranting an extra error term (ε_i) for each item.

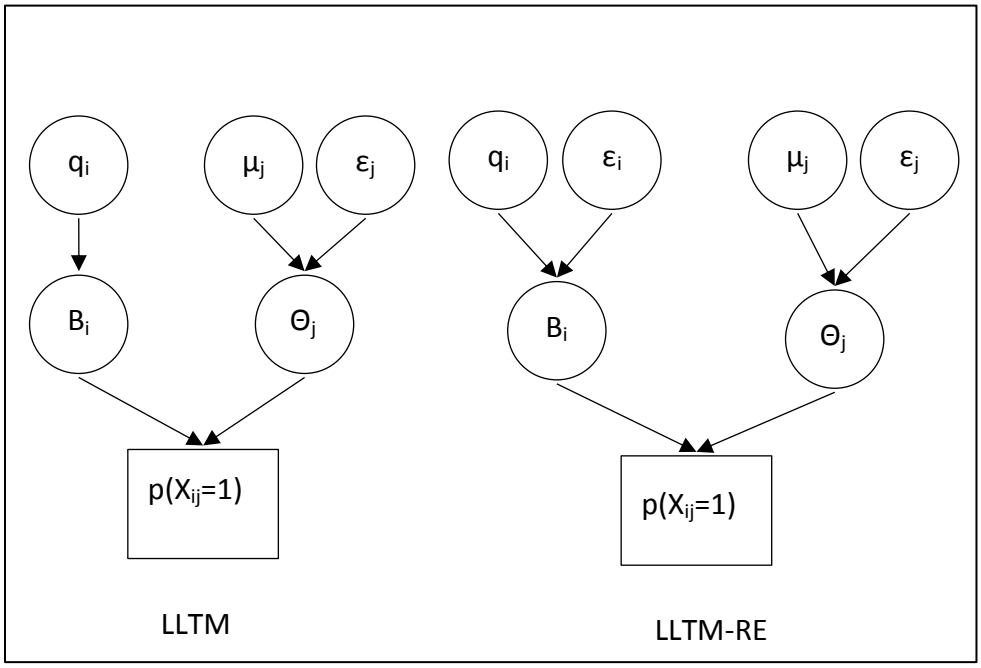

Figure 11. An illustration of the modeling differences between LLTM and LLTM-RE from Janssen (2010).

In short, the LLTM-RE implies there is a level of variance among all items that is extraneous to the effects explained by the cognitive features. The estimation of item difficulty can therefore be expressed as

$$\beta_i = \sum_{k=1}^{K} q_{ik}\eta_k + \varepsilon_i \, .$$

Since the item error is distributed equally among all items, this implies the item difficulty parameter for a given set of cognitive features falls under a distribution

$$\beta_i \sim N(\beta_i^*, \sigma_\epsilon^2),$$

where β_i^* is the item difficulty estimate from the LLTM model.

Few studies have applied the LLTM-RE to calibrate generated items in recent years. Arendasy (2010) demonstrated an application of LLTM-RE to calibrate generated items for spatial reasoning. Gierl and Lai (2011) demonstrated an application in elementary mathematics. Kubinger(2008) applied the LLTM-RE in an application to calibrate items written systematically. Lai and Gierl (2012) also described the required test design features to calibrate generated items using the LLTM-RE.

Summary

In this chapter, I provided a review of item generation according to three temporal phases: before, during, and after item generation. Researchers have conceptualized and improved on how information can be modelled for item generation. Technological innovations have been described that help us with the generation process itself. Statistical modeling methods have been proposed to estimate the psychometric properties of the items after they are generated.

Together, these three stages help us understand the current state-of-the-art in AIG.

Limitations in Current Approaches to AIG

Despite research on different stages of item generation occurring in different fields of studies, item generation has remained an esoteric approach to item development with few operational applications. In other words, researchers and practitioners tend to agree that AIG is a great idea, but the methods have rarely been used to create operational test items. One reason for the lack of implementation may be traced to the dearth of guidance on the generation process itself. As highlighted in my review, studies in item generation tend to focus on the psychometric aspects of AIG at the expense of content-related issues (e.g., creating item models), focus on content-related issues while neglecting the required item generation technology, or use technology without considering content or psychometric issues. An item generation framework is needed to consolidate and componentialize the required tasks starting from the first step of conceptualizing the content to the last step of generating operational items. Currently, no framework exists.

Item quality, from the current content specialists approach, is established using item writing guidelines and through standards of practice. Unfortunately, no comparable guidelines or standards exist for creating the item models capable of generating thousands of items. Although researchers have described the structure of the item model, no researcher has described the process of how

content expertise is captured in these item models. Moreover, with a preference for strong theory AIG, a cognitive modeling process designed specifically to facilitate item generation is needed to guide AIG practices. A dedicated cognitive modeling process can provide researchers and practitioners with a structured approach for identifying the information that should be extracted from content experts as well as how this information can be manipulated for item generation.

Template-based item generation is a systematic process for creating large numbers of test items. However, this AIG approach tends to yield items that are very similar to one another. Producing similar items limits the utility of AIG because items must be placed on test forms with diverse and variable content to increase test security and minimize item exposure. Template-based item generation, therefore, must be modified to ensure the generated items are more diverse.

The purpose of my dissertation research is to address these limitations by presenting and demonstrating a framework for AIG. The framework, named Systematic Item Generation (SIG), specifies how information can be identified and how item models can be used to generate large numbers of diverse items.

Chapter III: Systematic Item Generation

In this chapter, I present my proposed framework for item generation. This framework is intended to improve on current practices as well as provide AIG practitioners with concrete guidelines and examples for generating items. The framework is divided into three development stages: cognitive modeling, item modeling, and item generation. It should be noted that item calibration, as described in the previous chapter, will not be included in my framework. I was not able to collect sufficient information for calibration. This limitation will be further discussed in the final chapter of my dissertation.

Systematic Item Generation

As argued in Chapter 2, a new perspective is needed on how to generate test items. I contend a new item generation framework should be developed, not for the sake of technological demonstration or statistical purposes, but rather to address a real need for practical guidance in item generation. By addressing this need, content specialists and test developers will participate and contribute to AIG in a more meaningful way. Currently, much of the research on AIG is conducted by psychometricians who tend to focus on the statistical modeling or computing scientists who focus on the technological applications. My framework is intended fill this void by appealing to content specialists and test development practitioners. Further, item generation approaches should be robust to address different needs across different content areas, item types, or purposes of testing. To address the need for a generalizable and applicable

approach in generating items, I present a framework called systematic item generation (SIG).

Few examples exist in the literature on demonstrating how items can be generated (Gierl & Haladyna, 2012; Irvine & Kyllonen, 2002). When examples are presented, they are often constrained to a specific domain and developed to address a specific test development goal. Never has a generalized framework been presented or described to facilitate item generation. SIG is an attempt to provide a generalized framework for item generation that can be used for any content area. SIG can be conceptualized as a three-stage development process that garners information to support a manufacturing process to produce test items. The idea of a manufacturing process follows an assembly line model of systematic development, where dependent components are combined to produce a final product. In this manufacturing process, each development stage takes information in one state, manipulates and appends new data, and transforms it into a new state of information just as one might expect as a product created using an assembly line. Each stage of development is dependent on the outcome from the previous step. Figure 12 is an illustration of the processes of SIG. Detailed descriptions on each stage of the SIG framework are presented next.

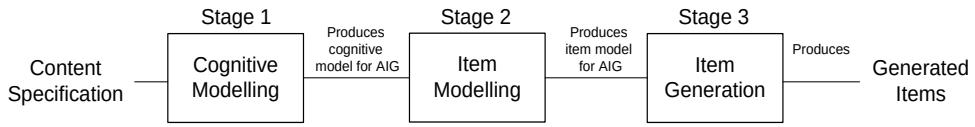

Figure 12. An illustration of the SIG process.

Stage 1: Cognitive Model Development

Although item writing guidelines are widely available, few instructions guide content specialists on how items can be generated and what type of information should be included to promote this generation process. Unlike item writing, where content can be reviewed for each item, generating large numbers of test items require the content be expressed collectively by content experts. The first stage of SIG, also known as cognitive model development for AIG, involves translating item specifications into a concrete structure of knowledge that can be used to produce new items. That is, content expert knowledge is extracted from the item specification to form a cognitive model that will be used to create new items.

Test items are developed to address the demand from, and demand to fulfill, requirements of a test specification or content domain blueprint. Therefore, content specifications could be used as a starting point to form a structure of knowledge to organize and constrain the content for item generation. Defining this structure is a challenging task, as content experts must narrow a large amount of relevant information into a knowledge structure while

ensuring numerous possible outcomes can still be used to generate a diverse set of items.

To address these complexities in modeling knowledge for item generation and provide guidance to content experts required to create such models, I introduce a knowledge structure to facilitate the cognitive modeling of content expertise used for systematic item generation (SIG)[1]. Current item generation methods document the structure of the item model, but how such models can be created and what type of information content experts provide is not yet documented. Using a cognitive modeling process that is designed specifically to facilitate item generation, this process could provide a structured approach to guide what type of information will be extracted from content experts and how their knowledge can be manipulated for item generation.

The knowledge structure I propose follows the abstraction-decomposition representation often used for representing expert knowledge (Hoffman & Lintern, 2006). The SIG knowledge structure consists of three parts: problem and scenario, sources of information, and features. Problem and scenario describes the overall issue that all generated items should probe. The problem is a general description, while scenarios are specific descriptions of the content related to the overall problem. Together, the two components provide the broad coverage needed for an item model as well as the specificity of

[1] In this study, I refer to the format of the cognitive model as a knowledge structure, and I refer to it as a cognitive model when the structure is filled with content knowledge.

content needed for each item. Sources of information organize the cues and features that relate to the problem. Information source can be either general or specific to the problem, meaning whether an information source is varied for the sake of isomorphism (general) or varied conditionally on the presented scenario (specific). The information source is used to organize knowledge for content experts, along with providing an additional layer of information needed for item modeling. Moreover, as features are categorized into different information sources, it is also used to prompt content experts to think of other sources that may be related to the presenting problem. Features are a set of clues or characteristics that are categorized under an information source. For any cognitive model, multiple features may be presented under a single source of information, each feature may be presented differently depending on the presenting scenario, occurrence of other features, or other conditions. This condition for each feature is expressed by content experts in the form of constraints. Figure 13 is an illustration of the complete knowledge structure.

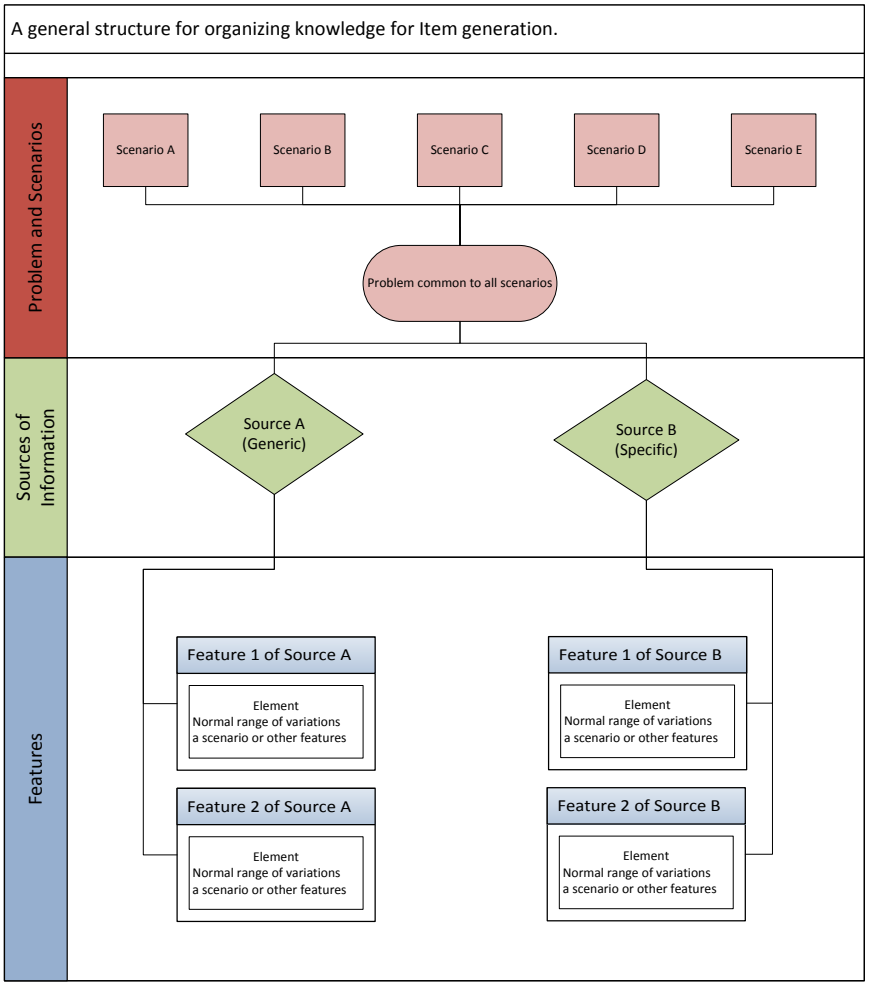

Figure 13. An illustration of knowledge structure for item generation.

Using this knowledge structure, content experts are required to express knowledge in a systematic manner through an interview process to produce a cognitive model for AIG. To start, content experts are asked to either define an overall problem that an item model should probe or a content specification that must be presented. Then, content experts describe the features required to solve each problem. These features have to be either relevant to the presenting scenario or include information that differs between scenarios. The features are grouped into sources of information, as expressed by the content experts, where

information sources prompt content experts to provide more features to a source or create more sources. Finally, experts define the relationships between scenarios and features. These relationships are expressed as constraints in each feature.

As knowledge is provided by content experts, this knowledge structure serves as a guide to ensure sufficient information has been provided to promote item generation. This approach also overlaps with much of the current research in educational assessment, as there has been a renewed focus on the use of cognitive models in test development (e.g., Leighton & Gierl, 2011). By outlining the knowledge, skills, and processes required to solve a problem in the form of a cognitive model, items can be generated based on the knowledge provided by content experts in this manner.

Stage 2: Item Model Development

In this step, information from the cognitive model is transformed into an item model. This stage is needed to ensure all components defined in the cognitive model are represented in an item model. Hence this second stage of SIG is known as item model development. Recall, an item model is a prototypical representation of a test item that guides the generation process. Examples of item models were presented in the last chapter and available in the literature (e.g., Bejar et al., 2003; Case & Swanson, 2002; Gierl et al., 2008). In contrast to cognitive model development which focuses on incorporating information from

content experts, item model development is a systematic process that incorporates information from the cognitive model into a modular format suitable for item generation. Lai, Gierl, and Alves (2010) demonstrated this systematic process of item model development using the assessment engineering approach to test design. In the Lai et al. study, the demands of an assessment task were stated in the form of a first-order prose and item models were created by incorporating each demand in the prose into the item model iteratively using computer technology.

Item models provide the prerequisite knowledge presented in the form of a cognitive model which is then transformed into a different knowledge state to provide a prototypical representation of a test item. This representation, once created, will then lead to operational item generation in stage 3. However, current item modeling techniques are often criticised for leading to generated items that appear too similar to one another. To address this item modeling concern, I will propose and illustrate an alternative type of item model called an *n-layer item model* to allow for more text variability among the generated items thereby yielding heterogeneity in the item generation process.

N-layer Item Model

Cloning, in a biological sense, refers to any process where a population of identical units is derived from the same ancestral line. Current approaches to generating items from models as described in the last chapter often yield

outcomes that are described by content specialists and test developers as "cloned" or "ghost" items. The resulting similarity between generated items is due, in large part, to the number of elements in an item model which, typically, is limited by the word length of the item stem. This limit on the number of elements has rendered current approaches to item generation as fixed and finite leading to item cloning outcomes. Clones are perceived by content specialists to be generated items that are easy to produce, unlike more traditional items. Clones are often seen as a simplistic product from an overly simple item development process, compared to a more sophisticated traditional test item which is a complex product from a more sophisticated item development process. Most importantly, clones are believed to be easily recognized by coaching and test preparation companies which limit their usefulness in operational testing programs. As a result, content experts are rarely impressed with items produced from template-based AIG approaches, particularly when the underlying model is thought to be discernible through the generated items. While this restriction is inherent to the use of elements for item generation, the construct of the element can be changed to accommodate more variation. In this study, I refer to current item models as a *1-layer item model*, where an item model only contain elements as they are presented. By conceptualizing elements as objects that can be embedded within other elements, a new type of item model can be created. This new approach, called *n-layer item model,*

allows more elements to vary in an item model where manipulations can occur under two or more layers of elements.

Much like the 1-layer item model, the starting point for the n-layer model is from a parent item. But unlike the 1-layer model where the manipulations are constrained to a linear set of generative operations using a small number of elements at a single level, the n-layer model permits manipulations of a nonlinear set of generative operations using elements at multiple levels. As a result, the generative capacity of the n-layer model is substantially increased. A comparison of elements used in 1- and n-layer item models are presented in Figure 14. In this example, the elements in a 1-layer model can provide a maximum of four different values for element A. Conversely, the n-layer model can provide up to 64 different values using the same four values for elements C and D embedded within element B. Because the maximum generative capacity of an item model is the product of the ranges in each element (Lai, Gierl, & Alves, 2010), the use of an n-layer item model will always increase the number of items that can be generated relative to the 1-layer structure.

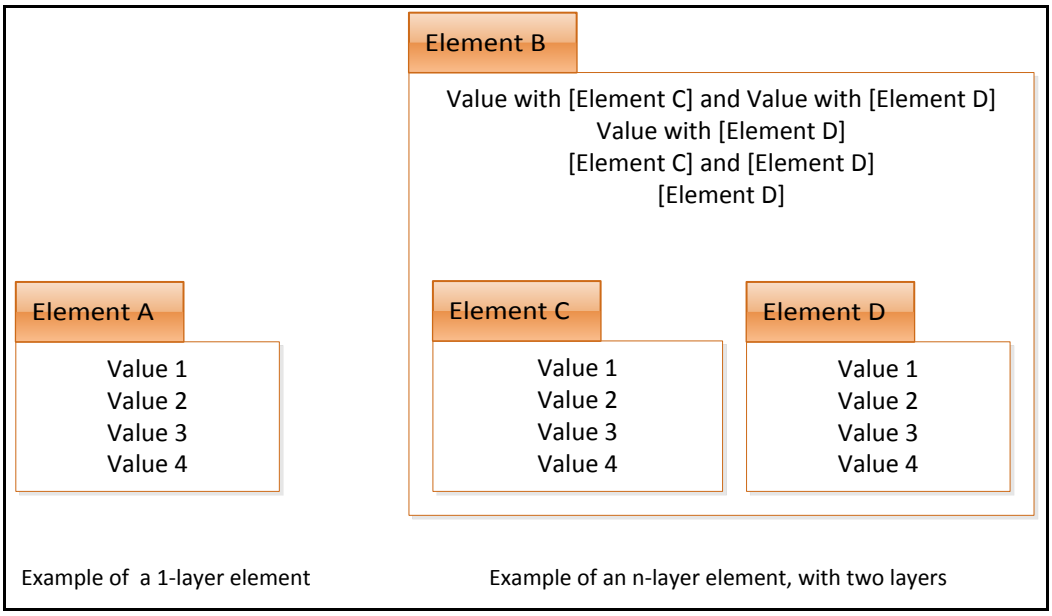

Figure 14. A comparison of the elements in a 1-layer and n-layer item model.

The concept of n-layer item generation is adapted from the literature on syntactic structures of language where researchers have reported that sentences are typically organized in a hierarchical manner (e.g., Higgins, Futagi, & Deane, 2005). This hierarchical organization, where elements are embedded within one another, can also be used as a guiding principle to generate large numbers of meaningful test items. The use of an n-layer item model is therefore a flexible template for expressing different syntactic structures, thereby permitting the development of many different but feasible combinations of embedded elements. In the natural language processing literature, our n-layer structure could be characterized as a generalized form of template-based natural language generation, as described by Reiter (1995).

In the past, item models are created directly by content experts for item generation. As item model development required substantial computer programming knowledge as well as an in depth knowledge in the item generation capabilities, developing item models became a fruitless process for most content experts resulting in models that produced relatively few items (Lai, Alves, Zhou, & Gierl, 2009). By developing item models systematically from the cognitive model contents, minimal input is needed from content experts during this stage. The resulting item models enable the next stage of development, item generation.

Stage 3: Item Generation

The third stage of SIG involves how item models can be translated and extracted in the form of test items. With item models produced from stage two, items can now be generated by iterating all unique combination of elements into new items. The generation process for SIG requires two functions: the ability to iterate all unique combination of elements expressed in the item model and the ability to evaluate whether a specific combination of elements meet the requirements of constraints in the item model. Based on these two requirements, the IGOR program, as described previously in Chapter 3, is used to demonstrate the generative stage of SIG in this study.

Summary

The purpose of this chapter was to introduce the Systematic Item Generation (SIG) framework. The three stages of SIG framework--cognitive modeling, item modeling and item generation--provide a guide for generating test items. In addition, methods were also introduced to ensure test developer demands for test items were being met. The knowledge structure for item generation was introduced to anchor content knowledge throughout the generation process. The n-layer item model was introduced to improve item modeling approaches and increase text variability between generated items. By conceptualizing item generation into a three-stage manufacturing process that moulds assessment knowledge into different states, the SIG framework can be applied to provide insights into how item generation can be realized in different content areas.

However, with the focus of the SIG framework aimed at improving the quality of generated items, the scope of this framework is limited to describing the development process from extracting content expert knowledge to generating test items. As a result, issues related to item generation such as the integration of the generation framework with the test design (i.e., curriculum or assessment system alignment), or generated item calibration are beyond the scope of this framework and this study. In the next chapter, I apply the methods to my proof-of-concept application of the SIG framework in the medical

education domain and demonstrate how items can be generated from content expert knowledge.

Chapter IV: Application

With an introduction to Systematic Item Generation (SIG), I now present a proof-of-concept application. A proof-of-concept is the operationalization and implementation of a specific method—for my dissertation, this method is SIG—to demonstrate its potential and feasibility for actual use. In this chapter, I use SIG to generate test items in the domain of medical education with data and information collected through my collaborative work with the Medical Council of Canada. The chapter is presented in two sections. First, I provide some background information on the content area of this demonstration as well as an overview of cognitive modeling in the field. Then, I present a demonstration of the SIG framework applied to generate test items in the surgical content areas of hernia and post-operation fever. In presenting this application, I demonstrate how items can be generated from content expert knowledge and illustrate the three stages necessary for item generation. The outcome from my proof-of-concept demonstration is the production of new surgical test items.

Medical Council of Canada Qualifying Examination–Part 1

The Medical Council of Canada Qualifying Examination-Part 1 (MCCQE-1) is a one-day, fixed-length, multi-stage computer-adaptive test used to assess the knowledge, clinical-reasoning skills, and attitudes specified by the Medical Council of Canada as key objectives and competencies for medical training. Required by the Licentiate of Canada, the MCCQE-1 is written by all medical students seeking entry into supervised clinical practice for postgraduate training

programs in Canada. Section 1 of the MCCQE-1 is a 3.5-hour test containing 196 multiple-choice items administered adaptively by computer across six content areas. Section 2 of the MCCQE-1 is a 4-hour test containing clinical decision-making items. My demonstration is focused on generating multiple-choice items for section 1 of the exam.

Each year, approximately 4,000 candidates who have completed their medical training attempt the MCCQE-1 in one of two testing windows (spring and fall). The MCCQE-1 is composed of six different content areas: Internal Medicine, Obstetrics and Gynecology, Pediatrics, Psychiatry, Surgery, and Population Health. Items in each area are developed by a panel of content experts specializing in each respective content area. Prior to administration, items are classified into four levels of difficulty ranging from 1-easy to 4-difficult based on their psychometric properties. Then, items are assembled into four-item testlets according to the item difficulty estimates. In total, 196 items are selected and administered to each examinee across the six content areas using the multi-stage computer adaptive testing process.

Because the MCCQE-1 is administered in multiple sites across Canada with each administration window lasting up to one month, adaptive administration procedures are used to minimize exposure of test items to examinees. The MCCQE-1 is administered with seven stages of adaptation known as panels (Luecht, 2002). Test items are assembled into four-item testlets within each of the six content areas, then testlets from all six content areas,

along with a testlet of field test items, are administered to the examinee as a panel (see Figure 15). After a panel of items has been completed by the examinee, results are scored and used to determine the next best set of items given their performance. Seven panels of items are administered in this fashion for all examinees.

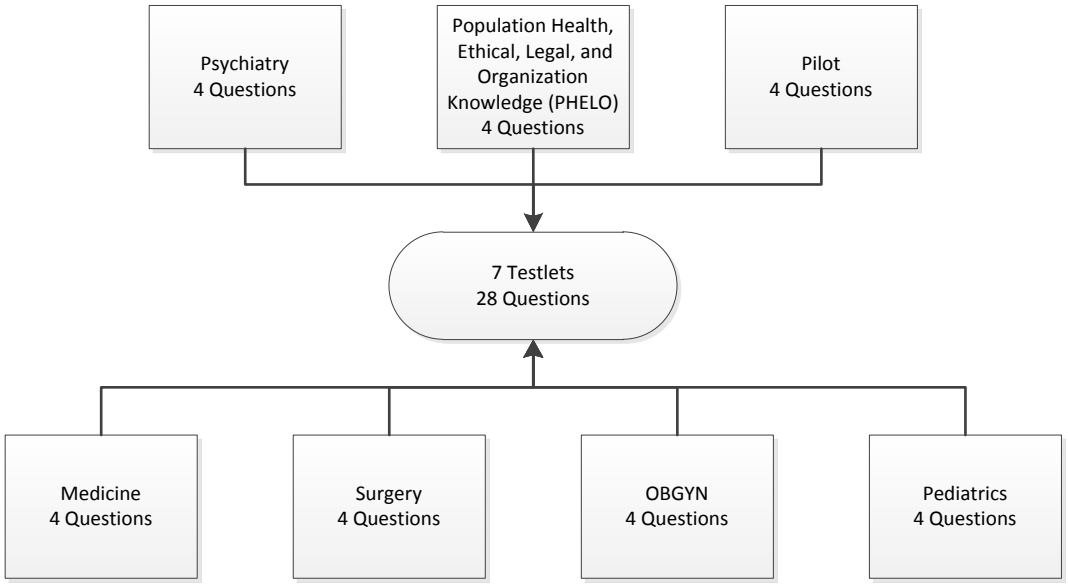

Figure 15. Composition of one panel of multiple choice items in the MCCQE-1.

Thousands of items are needed to operationalize computer adaptive tests. To enable the procedure used in MCCQE-1, the current exam contains a bank of 2,803 test items. These test items are developed using a traditional process of item writing (see Figure 16). First, demand for test items are identified based on the test specification and an analysis of the resources in the existing item bank. Second, a content expert within one of the six content areas is asked to write items based on these demands. To ensure items are developed

in a uniform manner and adhere to a high standard of quality, a workshop is administered prior to item writing as a way of reminding the content experts of common item writing strategies and techniques. Third, after item writing is completed, the newly created items are vetted and revised by a panel of experts. This revision process allows each item to be evaluated by different experts to ensure it meets the required specifications of content relevance and psychometric quality. Fourth, after items are vetted for content and format adjustments by various reviewers, items are field-tested using a small sample of examinees to garner some evidence on the item's psychometric properties. Items that are deemed to satisfy the psychometric demands are then inserted into the item bank for operational use.

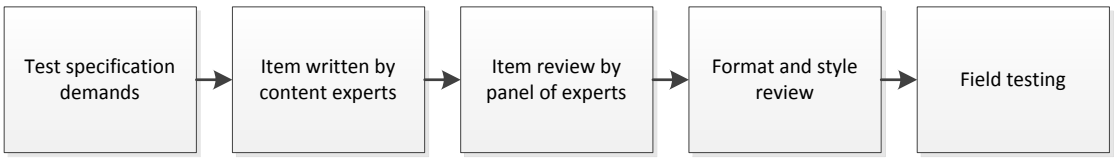

Figure 16. Current item writing process for MCCQE-1.

The current method of item development for the MCCQE-1 is resource intensive and time consuming. The content experts who develop items for the MCCQE-1 are also practicing physicians. Hence, coordinating sufficient time for an expert panel to create test items is often challenging. Moreover, item development is also limited by a lengthy review process, requiring every item to

be reviewed by each panelist. For example, if a panel contained six medical experts, then the number of items they create is limited by how many items they can collectively write and review as a group. While each content expert often creates 10 items in one day, time is also required for the panel to review all 60 items (if each experts creates 10 items). With relatively few items being developed from this time-consuming and expensive process, there is a constant demand for new MCCQE-1 items. The demand for items is further complicated when there are curricular changes that require the development of new items. Simply put, there is a high demand for item generation with the MCCQE-1.

Modeling Medical Education Knowledge

Knowledge, procedures, and practices exist for conceptualizing and modeling medical knowledge. Norman, Eva, Brooks, and Hamstra (2006) described different modeling approaches as causal (basic mechanisms of medicine), analytic (relationship of specific symptoms and features with specific conditions), and experiential (prior case experiences), where each type of modeling contributes unique knowledge to our understanding of how experts solve medical problems. For tests in medical education, knowledge is often framed in problem-solving aspects (Norman, 1988; Custers, Stuyt, & De Vries, 2000).

There are two common approaches for modeling medical problem solving. Diagnostic knowledge of a medical problem can be expressed in a

hypothetico-deductive manner, where physicians generate a hypothesis about a medical condition and collect information to support or refute the claim (Elstein, Shulman, & Sprafka, 1990). For instance, if a physician is presented with a patient who has a fever after surgery, then the physician will actively seek patient information to deduce the most probable diagnosis. Alternatively, medical problems can also be expressed in a schema-inductive manner. Coderre, Mandin, Harasym, and Fick (2003) claimed that under this method, a physician "seeks specific information from the patient … that will distinguish between the categories of conditions at the branching points on the scheme". In contrast to collecting patient information to deduce the correct diagnosis, the schema-inductive approach is driven by the physician's use of patient information to fit any diagnostic schemas that best describes the patient's scenario.

The two prevailing methods of conceptualizing medical knowledge parallel the confirmatory and exploratory dyad of knowledge representation. Norman and Eva (2003) suggested there is no perfect approach in medical problem solving and, as a result, experts may use multiple problem-solving approaches at the same time. To generate new test items that enable examinees to employ either problem-solving strategy, a third approach for modeling content knowledge can be described as pattern recognition.

Pattern recognition is a popular method used in machine learning. It is a stochastic approach for modeling learning where a given set of inputs consistently produces a set of expected outcomes, even when the rules that generate the expected outcomes are not explicit. In the context of medical problem solving, certain characteristics of a patient (input) often lead to the diagnosis of a specific medical condition (output). The cognitive modeling approach outlined in this study focuses on having content experts identify a set of patient features for a given medical problem. Pattern recognition then allows examinees to use features with different problem-solving strategies (e.g., inductive or deductive), but with a restrictive outcome where the presented features only yield one correct solution in the form of the keyed or correct multiple-choice item option. Coderre et al. (2003) claimed that experts often use a pattern recognition approach because it often yields a correct diagnosis. Pattern recognition from clinical features is also a method that can be used to present hints about the correct solution during medical problem solving (Kazi, Haddawy, & Suebnukarn, 2012).

In this study, it is imperative that the cognitive modeling approach in SIG is representative of the content areas presented on the MCCQE-1. But in addition to modeling knowledge as expressed by content experts, the cognitive models must also yield test items. In the next section, I demonstrate how content knowledge can be collected from experts to create a cognitive model for item generation.

Generating Items to Evaluate Medical Knowledge

In this section, I describe how medical knowledge can be identified from content experts and then used for generating test items. To begin, I demonstrate the first stage of SIG by extracting knowledge from content experts and placing this knowledge into cognitive models. Specifically, I present the procedures and processes that were required to extract information from content experts for creating cognitive models in the surgical areas of hernia and post-operative fever. Then, using the cognitive models, I demonstrate the second stage of the SIG and produce two item models, one for each cognitive model. Finally, I demonstrate the third stage of SIG by using IGOR to generate items from the two item models.

Stage 1: Cognitive Modeling of Context Expertise

Content specifications from the MCCQE-1 were used to identify the content for the generated items. Two content experts, who were both practicing surgeons and experienced medical item writers, provided their expertise. Knowledge from the content experts was captured using a semi-structured interview process. The content experts start by reviewing a set of existing test items to select two surgery applications. They selected hernias and post-operative fever. Their knowledge and skills required to diagnose problems in these two areas were then identified in an inductive manner, meaning the content specialists were given an existing multiple-choice item and asked to identify and describe the key information that would be used by an examinee to

diagnose a specific problem and, hence, solve the item. This knowledge was then expressed into a cognitive model. The relevant cognitive components and the associated interview questions posed to the content experts are presented in Table 1. From this extraction process, cognitive models in each content area were developed.

Table 1.

A list of questions used for extracting content expert knowledge into a cognitive model.

Relevant Component	Question
Problem	What is the overall problem this model is describing?
Scenarios	What are some of the different conditions or scenarios that may arise from this problem?
Features	What are some features of the patient that are typical of a patient with a given scenario?
Sources of Information	Can these features be organized under different sources of information?
Features	Are there any more features that are typical of patients under a given scenario?
Features	What are the relationships between each feature and scenario?
Features	Are there any relationships between each feature?
Scenarios	What is the key for these scenarios?
Scenarios	What are some distracters that are common to these scenarios?

Hernia

Cognitive modeling began with this parent item:

A 24-year-old man presented with a mass in his left groin. It appeared suddenly 2 hours ago while lifting a piano. On examination he has a tender firm mass in the left groin. Which one of the following is the next best step?

1. Immediate hernia repair
2. Needle aspiration
3. Ice packs to groin
4. Reduction of mass
5. Ultrasound of groin

From this item, the content experts summarized the problem specific to the item as surgical issues related to hernia. Then, the experts identified four different scenarios related to that problem. The identified scenarios were different subtypes of hernia: asymptomatic incarcerated, painful incarcerated, strangulated, and reducible symptomatic. These scenarios are considered to be possible outcomes of a hernia surgical problem. These scenarios are also representative of different issues related to hernia that will each be presented with a unique set of features (as described below). The problem and related scenarios are presented in the top panel of Figure 17.

After the problem and scenarios were identified, content experts identified 10 features that either vary across the different scenarios or are features that should be presented with the patient regardless of scenarios. Of the set of 10 features, seven are specific or vary by scenario (i.e., scenario

dependent), while three are general or vary regardless of scenario (i.e., scenario independent). To organize the 10 features expressed in the different scenarios, four sources of information were introduced to group these features. The four information sources include patient presentation, location, physical examination, and laboratory results. Next, patient characteristics that vary by scenario are expressed as constraints and are added to each corresponding feature. To identify and structure this information, the two experts first identified the constraints for each feature. For example, with the feature that describes pain, the nominal value for pain across all patients is that they have no pain. The patient only presents with pain in the event of a painful incarcerated (presents with intense to severe pain), strangulated (presents with severe pain), and reducible symptomatic (presents with moderate-mild pain) hernia. Then, the experts identified relationships in the model that needed to be accounted for in order to diagnose complications with hernias such as controlling co-occurrences of two features (e.g., only one of groin pain, umbilicus pain, and scars can be presented with the same patient), constraints between scenarios, and constraints between features. Using this approach, the relationship between each feature and scenario was identified. The resulting cognitive model is presented in Figure 17.

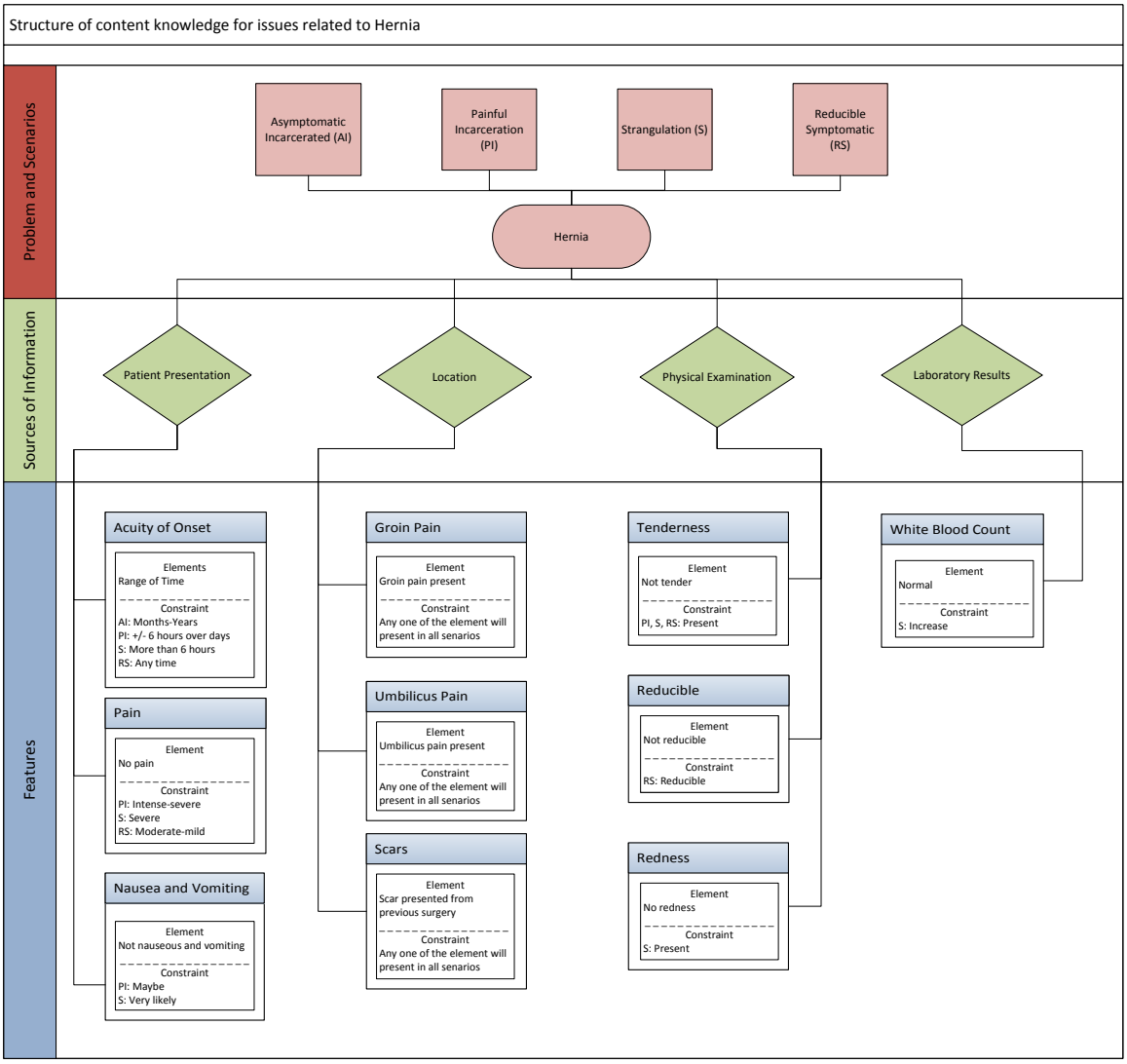

Figure 17. The cognitive model for generating items in hernia.

After the modeling of the item stem is complete, information about the key and distracters needed to provide options for the generated items are extracted. For the hernia example, each scenario is associated with a unique first-line treatment. That is, asymptomatic incarcerated hernia does not require anything beyond ice applied to the mass; symptomatic incarcerated hernia and reducible symptomatic hernia require reduction of hernia; and strangulated

hernia requires hernia repair. In addition to these hernia-specific options, exploratory surgery was identified as a good distracter that could be used with any one of the generated items. Collectively, the information expressed in the cognitive model can be used to generate stems and options for a range of hernia items.

Post-Operative Fever

The same process was used to create a cognitive model for generating items required to diagnose complications associated with post-operative fever. The content experts selected the following parent item as their starting point:

> **A 65-year-old man has a right hemicolectomy for cancer. On post-operative day 2 he has a temperature of 38.2 C. Which one of the following is the most likely diagnosis?**
>
> 1. Urinary tract infection
> 2. Atelectasis
> 3. Wound infection
> 4. Pneumonia
> 5. Deep vein thrombosis

From this item, the content experts agreed that post-operative fever was the problem described in the item. They identified six possible scenarios related to post-operative fever. These scenarios are urinary tract infection, atelectasis, wound infection, pneumonia, deep vein thrombosis, and deep space infection. From these six scenarios, the content experts identified 12 features that could be manipulated in test items. These features were age, gender, timing of fever, guarding and rebound, temperature of fever, abdominal examination, red and

tender wound, gastrectomy, left hemicolectomy, right hemicolectomy, appendectomy, and laparascopic cholestectomy. Of the 12 features, three (age, gender, and temperature) were generically varied while the remaining nine features were surgery specific.

After identifying all features, the experts then organized them into four sources of information: patient demographics, timing of fever, physical presentation, and type of surgery. To define the constraints for each feature, the experts identified the relationship of each feature to each scenario. The resulting cognitive model for post-operative fever is presented in Figure 18.

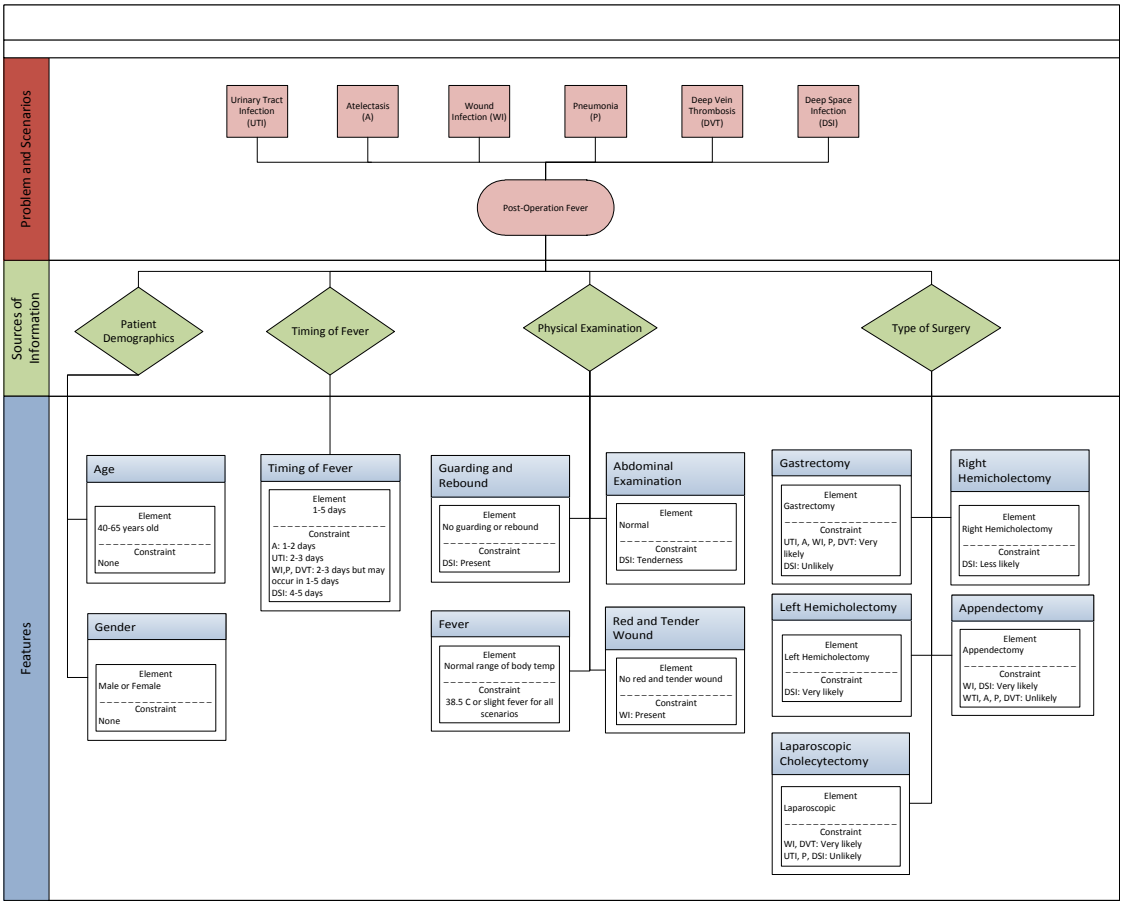

Figure 18. *Illustration of the cognitive model for generating items in post-operative fever.*

After modeling the item stem, the key and distracters were identified for all generated items. Similar to the hernia item model, options are needed to prompt examinees not to diagnose the scenario at hand, but to select the best next step in managing the scenario (i.e., select a plausible but incorrect distractor). The best management option for each scenario is presented in Table 2. For example, if post-operative fever is caused by a urinary tract infection, then the best management option would be to administer antibiotics. A combination of options from each scenario served as the distracters for this model.

Table 2.

Best patient management options for each scenario in post-operative fever model.

Scenario	Best management option
Urinary tract infection	Antibiotics
Atelectasis	Mobilize
Wound infection	Open wound
Pneumonia	Antibiotics
Deep vein thrombosis	Anti-coagulation medicine
Deep space infection	Drainage

In sum, the content experts described their conceptualization of hernia and post-operative fever. We use this conceptualization to identify and structure the content in the form of a cognitive model. The cognitive model used in this framework consists of three components: problem and scenarios, sources of information, and features. These components, in turn, will be used to generate test items. Next, I demonstrate the second stage of the SIG framework by describing how content from cognitive models can be converted into an item model structure.

Stage 2: Item Modeling

In the second stage of SIG, the information captured in the cognitive model guides the creation of the item models. By incorporating each feature from the cognitive model into an item model using a systematic process, items can be generated (Lai, Gierl, & Alves,2010). Each feature in the cognitive model is incorporated into the item model as an element. Next, the scenarios and

information related to the options are incorporated into the item model. Last, the constraint for each feature is organized and expressed in the item model to prevent the generation of undesired or nonsensical items. In other words, I demonstrate how the two cognitive models from the first stage can be transformed into item models as part of the second stage. In addition to the traditional 1-layer item model, this process can also be used to demonstrate my new n-layer item modeling approach as part of the proof-of-concept for my dissertation research.

1-Layer Item Models

Hernia

The starting point for generating hernia items is to append the ten features outlined in the cognitive model into the item model. The resulting stem with features is presented in Figure 19. Because information across features can sometimes be presented together, some features from the cognitive model in this example are collapsed and expressed as one element in the item model. For example, in the physical presentation of the patient, all features are varied under different scenarios. The three features of physical presentation are therefore expressed as one element in the item model to reduce model complexity. Also, the location of the pain varies randomly in this example. Therefore, the different location features were collapsed and expressed as one element that varies depending on the scenarios. Finally, features about pain, nausea, and vomiting

were collapsed into one element to simply the item model. The resulting item model for hernia is presented as five elements. The content for each element is presented in Figure 19.

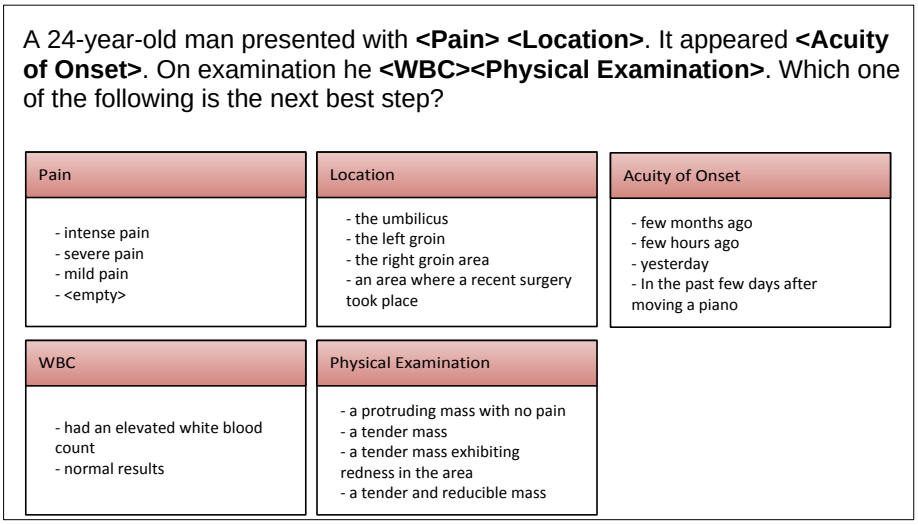

Figure 19. Stem of hernia item model with features appended.

After the stem is created, options are incorporated into the item model. In this example, four options were used. These options include ice applied to mass, hernia repair, reduction of mass, and exploratory surgery. To ensure the item models generated plausible items, a constraint was added to restrict the presentation of physical examination, level of pain, and acuity of onset with the corresponding scenario for each item. For example, when the scenario for an item is related to strangulated hernia, a patient should not be presented with mild pain or with a mass that is reducible. The complete 1-layer item model is presented in Figure 20.

Stem	A [AGE]-year-old [GENDER] presented with a mass [PAIN] in [LOCATION]. It occurred [ACUITYOFONSET]. On examination, the mass is [PHYSICALFINDINGS] and lab work came back with [WBC]. Which of the following is the next best step?
Elements	[AGE] (Integer): From 25.0 to 60.0, by 5.0 [GENDER] (String): 1: man 2: woman [PAIN] (String): 1: 2: and intense pain 3: and severe pain 4: and mild pain [LOCATION] (String): 1: the left groin 2: right groin 3: the umbilicus 4: an area near a recent surgery [ACUITYOFONSET] (String): 1: a few months ago 2: a few hours ago 3: a few days ago 4: a few days ago after moving a piano [PHYSICALFINDINGS] (String): 1: protruding but with no pain 2: tender 3: tender and exhibiting redness 4: tender and reducible [WBC] (String): 1: normal results 2: normal results 3: elevated white blood cell count 4: normal results
Options	exploratory surgery; reduction of mass; hernia repair; ice applied to mass

Figure 20. 1-Layer Hernia item model.

Post-operative Fever

The item stem with features for the post-operative fever example is presented in Figure 21. Similar to the item modeling process for hernia, some features were combined and expressed as one element. For example, the three features from the timing of fever are expressed under one element. Also, the five features describing the type of surgery are combined and expressed as one element to simplify the item model. Finally, different types of physical

examination results are combined and expressed as one element. This truncation results in an item model with five elements.

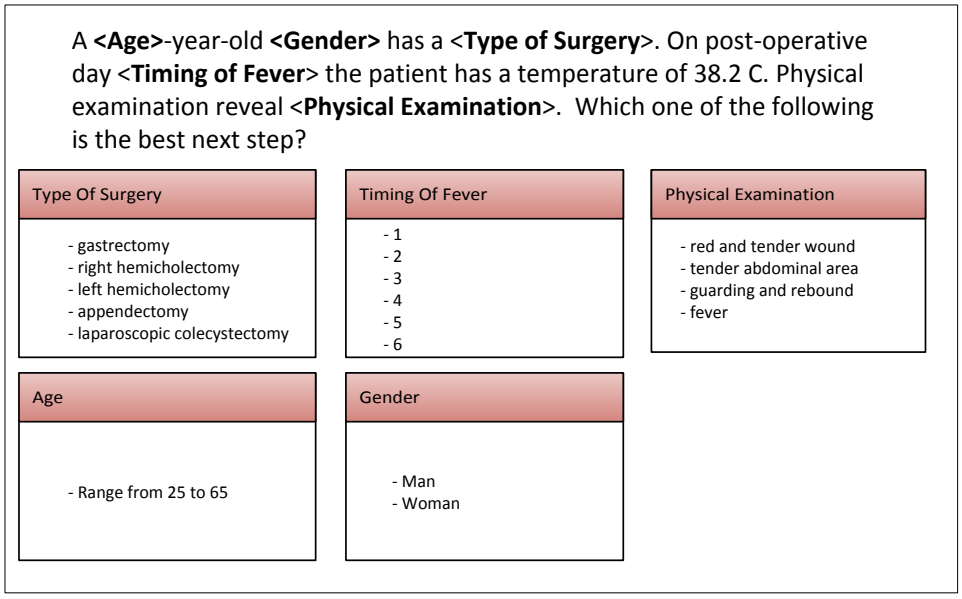

Figure 21. Stem of an item model on post-operative fever with features appended.

After the stem has been modelled, options must be identified. With six different scenarios highlighted in the cognitive model, five unique management outcomes can be identified and used as distractors. These five outcomes include antibiotics, mobilize, open wound, anti-coagulation medicine, and drainage. To ensure all generated items are plausible, constraints are defined in the item model. Three constraints were used with this item model. First, the timing of fever needs to occur within the expected timeframe for each scenario. Second, the procedure must be plausible for the given scenario. Third, the physical examination findings should be plausible for the given scenario. The

programming expression of the constraints are described in the next stage. The final 1-layer item model for post-operative fever problems are presented in Figure 22.

Stem	A [Age] year-old [Gender] was readmitted to the hospital for pain in the abdominal area. He was on post-operative day [Timing of Fever] after recovering from a [Type of Surgery]. The patient has a temperature of 38.5 C. Physical examination reveal [Physical Examination] Which one of the following is the best next step for this patient?
Elements	[Age] (integer) : 40 to 65 years old [Gender] (string): male or female [Timing of Fever] (string): 1 to 6 days [Type of Surgery] (string): Gastractomy, Right Hemicholectomy, Left Hemicholectomy, Appendectomy, Laparoscopic Cholecystectomy [Physical Examination] (string): Red and tender wound, Guarding and rebound, tenderness in the area
Options	Antibiotics, Mobilize, Reopen wound, Anticoagulation, Drainage

Figure 22. 1-Layer Post-Operative Fever Item Model.

N-Layer Item Modeling

N-layer item modeling helps address the problem of generating item clones by incorporating multiple layers of manipulation within an item model thereby resulting in more heterogeneity during the item generation process. But to convert the 1-layer item models into an n-layer item model, information must be reorganized in the item model. For 1-layer models, elements are composed of a list of values that can be replaced and substituted within the model. As a

result, each value within an element can be checked to ensure it fits with the desired item. With n-layer models, elements are embedded within one another resulting in a more complex verification process for each element. For example, when an n-layer item model is created, elements must fit not only with the desired item at one level, but also with items generated at different layers in the model. Because of the flexibility, but also complexity, associated with n-layer item modeling, a structure is needed to provide guidance on how n-layer item models can be realized to generate items. Fortunately, the SIG framework helps guide n-layer item modeling. Cognitive models organize features by the source of information. As a result, these information sources form a secondary layer to help organize groups of elements presented together. For example, sentences written in the item stem include a set of specific features. These sentences can then be manipulated as a secondary layer of elements. Moreover, different expressions of the same sentence allow for more variation in the item stem during generation. Next, I demonstrate how 1-layer item models can be converted into n-layer item models for generating hernia and post-operative fever items.

Hernia

The 1-layer item model presented in Figure 20 was expanded into an n-layer item model. From the 1-layer model, the three sentences expressed in the item stem were conceptualized as a second layer of elements. The last sentence

of the item stem can be used to demonstrate how sentences become elements. "Which of the following is the next best step?" is a question prompt that did not contain elements in the 1-layer case. But this question prompt can be reworded in the n-layer case and, as a result, be presented in different ways. That is, in the n-layer item model, the question is defined as an element, where two phrases: "Which one of the following is the best diagnosis?" and "Given this information, what is the best course of action?" serve as a new question element containing three values, where each value contains a question prompt. Different sentence structures can also be used to present patient findings. Table 3 describes four alternative sentences that can be used to present the same information. Within these sentences, three elements are embedded.

Table 3.

A table of values in the element of Patient Findings.

Original Patient Findings	Element on Patient Findings
On examination, the mass is [PHYSICALFINDINGS] and lab work came back with [WBC].	1. On examination, the mass is [PhysicalFindings] and lab work came back with [WBC]. 2. Upon further examination, the patient had [WBC] and the mass is [PhysicalFindings]. 3. With [WBC] and [PhysicalFindings] in the area, the patient is otherwise nominal. 4. There is [PhysicalFindings] in the [Location] and the patient had [WBC].

The first sentence in the stem can also be modified and presented as an element. In this sentence, five elements (age, gender, collated, pain, and acuity of onset) provide context for the examinee. But four alternative expressions can also be used to present the same information (i.e., elements), but in different manner. The values for this element are summarized in Table 4. Taken together, three elements at the sentence level form an additional layer in the item model. The first layer of elements manipulates the sentence structure while a second layer of elements manipulates the features as defined in the cognitive model within each new sentence structure. The complete n-layer hernia item model is shown in Figure 23.

Table 4.

A table of values in the situation element.

Original Situation	Element on Situation
A [AGE]-year-old [GENDER] presented with a mass [PAIN] in [LOCATION]. It occurred [ACUITYOFONSET].	1. A [AGE]-year-old [Gender] presented with a mass [Pain] in [Location]. It occured [AcuityofOnset]. 2. Patient presents with a mass [Pain] in [Location] from [AcuityofOnset]. The patient is a [AGE]-year-old [Gender]. 3. Patient complaints of a mass [Pain] in [Location] which has been a problem since [AcuityofOnset]. 4. A [Gender] [AGE] years of age was admitted with pain in the [Location] from [AcuityofOnset].

Stem	[Situation] [TestFindings] [QuestionPrompt]

Elements:

Layer 1	QuestionPrompt (Text): 1: What is the best next step? 2: Which one of the following is the best prognosis? 3: Given this information, what is the best course of action? TestFindings (Text): 1: On examination, the mass is [PhysicalFindings] and lab work came back with [WBC]. 2: Upon further examination, the patient had [WBC] and the mass is [PhysicalFindings]. 3: With [WBC] and [PhysicalFindings] in the area, the patient is otherwise nominal. 4: There is [PhysicalFindings] in the [Location] and the patient had [WBC]. Situation (Text): 1: A [AGE]-year-old [Gender] presented with a mass [Pain] in [Location]. It occured [AcuityofOnset]. 2: Patient presents with a mass [Pain] in [Location] from [AcuityofOnset]. The patient is a [AGE]-year-old [Gender]. 3: Patient complaints of a mass [Pain] in [Location] which has been a problem since [AcuityofOnset]. 4: A [Gender] was admitted with pain in the [Location] from [AcuityofOnset].
Layer 2	[AGE] (Integer): From 25.0 to 60.0, by 5.0 [GENDER] (String): 1: man 2: woman [PAIN] (String): 1: 2: and intense pain 3: and severe pain 4: and mild pain [LOCATION] (String): 1: the left groin 2: right groin 3: the umbilicus 4: an area near a recent surgery [ACUITYOFONSET] (String): 1: a few months ago 2: a few hours ago 3: a few days ago 4: a few days ago after moving a piano [PHYSICALFINDINGS] (String): 1: protruding but with no pain 2: tender 3: tender and exhibiting redness 4: tender and reducible [WBC] (String): 1: normal results 2: normal results 3: elevated white blood cell count 4: normal results

Options	exploratory surgery; reduction of mass; hernia repair; ice applied to mass

Figure 23. N-layer Hernia Item Model.

Post-operative Fever

To create an n-layer item model for generating items on post-operative fever, the 1-layer post-operative fever item model was modified. The item stem of the 1-layer model was reconceptualised in two layers of elements. In the first layer, the sentence structures of the stem are manipulated. That is, each of the three sentences presented in the item stem is modified to form an element. The modifications made to the first two sentences in the item stem are summarized in Table 5. Variation of the third sentence demonstrate how n-layer item models can be used to change generated test items. With the 1-layer item model, the prompt can only be used to ask examinees to choose the best next step given the information in the item. But with n-layer modeling, different questions can be presented to examinees with this prompt. In other words, by adding the sentence "Which one of the following is the most likely diagnosis for this patient?", the n-layer item model can now generate items to probe students on management strategies and diagnoses of post-operative fever issues in addition to generating items that require examinees to select the next best step. The final n-layer stem for the item model is presented in Figure 24.

Table 5.

Values of the situation and physical examination element.

Element	Original sentence	Values of the element
Situation	A [Patient Demographic] was readmitted to the hospital for pain in the abdominal area. He was on post-operative day [Timing of Fever] after recovering from a [Type of Surgery]. The patient has a temperature of 38.5 C.	1. A [Age]-year-old woman has a [Surgery]. On post-operative day [TimingOfFever] she has a temperature of 38.5 C. 2. A [Age]-year-old patient who had a [Surgery] had a fever of 38.6 C on post-operative day [TimingOfFever].
Physical Examination	Physical examination reveal [Physical Examination]	1. Physical examination reveal a red and tender wound at the opening. 2. Physical examination reveal tenderness in the abdominal region with guarding and rebound.

Stem	[Situation] [PhysicalExamination] [Question]
Elements:	
Layer 1	Question (Text): 1: Which one of the following is the most likely diagnosis for this patient? 2: Which one of the following is the best next step for this patient? Situation (Text): 1: A [Age]-year-old woman has a [Surgery]. On post-operative day [TimingOfFever] she has a temperature of 38.5 C. 2: A [Age]-year-old patient who had a [Surgery] had a fever of 38.6 C on post-operative day [TimingOfFever]. PhysicalExamination (Text): 1: 2: Physical examination reveal a red and tender wound at the opening. 3: Physical examination reveal tenderness in the abdominal region with guarding and rebound.
Layer 2	Age : 40 to 65 years old Timing of Fever: 1 to 6 days Type of Surgery: Gastractomy, Right Hemicholectomy, Left Hemicholectomy, Appendectomy, Laparoscopic Cholecystectomy Physical Examination: Red and tender wound, Guarding and rebound, tenderness in the area

Figure 24. Stem of the n-layer post-operative fever item model.

Because the n-layer model can now be used to generate items that require examinees to either make a diagnosis or provide the best management options, the key and distracters of the generated items must be adjusted accordingly to match the questions from the stem. As illustrated in Figure 25, the first layer of distracters present elements that either provide keys or distracters to either the diagnosis or management prompt. That is, the key of the item model contains two values, one value for the diagnosis prompt (*Key.diag*) and one value for the management prompt (*Key.man*). The second layer of elements for the distracters describe the content for each item. For

example, the element *Key.man* contains the management key for each corresponding scenario (e.g., antibiotics, mobilize, reopen wound, anticoagulation medicine, drainage) and the element Key.diag contains the diagnosis key for each corresponding scenario (e.g., urinary tract infection, atelectasis, wound infection, deep vein thrombosis, deep space infection). Using this coding approach, the distracters are defined for their corresponding item types. With the four item models created from the hernia and post-operative fever cognitive models, items can now be generated in Stage 3.

Layer1	a. Key (Text): 1: [Key.diag] 2: [Key.man] b. Option1 (Text): 1: [Option1.diag] 2: [Option1.man] c. Option2 (Text): 1: [Option2.diag] 2: [Option2.man] d. Option3 (Text): 1: [Option3.diag] 2: [Option3.man]
Layer2	Option3.man (Text): 1: Antibiotics 2: Mobilize 3: Reopen wound 4: Antibiotics 5: Anti coagulation 6: Drainage Option3.diag (Text): 1: Urinary tract infection 2: Actelectasis 3: Wound infection 4: Pneumonia 5: Deep vein thrombosis 6: Deep space infection Option2.man (Text): 1: Antibiotics 2: Mobilize 3: Reopen wound 4: Antibiotics 5: Anti coagulation 6: Drainage Option2.diag (Text): 1: Urinary tract infection 2: Actelectasis 3: Wound infection 4: Pneumonia 5: Deep vein thrombosis 6: Deep space infection Option1.man (Text): 1: Antibiotics 2: Mobilize 3: Reopen wound 4: Antibiotics 5: Anti coagulation 6: Drainage Option1.diag (Text): 1: Urinary tract infection 2: Actelectasis 3: Wound infection 4: Pneumonia 5: Deep vein thrombosis 6: Deep space infection Key.man (Text): 1: Antibiotics 2: Mobilize 3: Reopen wound 4: Antibiotics 5: Anti coagulation 6: Drainage Key.diag (Text): 1: Urinary tract infection 2: Atelectasis 3: Wound infection 4: Pneumonia 5: Deep vein thrombosis 6: Deep space infection

Figure 25. Options of the n-layer post-operative fever item model.

Stage 3: Item Generation

In the third stage of SIG, items are generated. Software is used to assemble all permissible combination of elements, subject to constraints articulated in the cognitive model. IGOR, or **I**tem **G**enerat**OR**, is a JAVA-based program designed for this assembly task. IGOR iterates through all possible combinations of elements and options. Without the use of constraints, all of the variable content (i.e., values for the integers and strings) would be systematically combined to create new items. Unfortunately, some of these items would not be sensible or useful. Hence, constraints serve as restrictions that must be applied to the assembly task so that meaningful items are generated. In this section, I describe how item models can be programmed and generated in IGOR. Then, I demonstrate the generative outcomes from the hernia and post-operative fever models.

Programming Item Models

To generate items from IGOR, item models are entered through a user interface organized with the same structural labels as the model. The stem section defines the content and programming required to create the item stems. Hypertext mark-up language (HTML) is used to format the stem. For IGOR to recognize an element, double square brackets distinguish the elements from other parts of the stem. Elements in IGOR can be defined as a numerical range

or a list of text variations. Similarly, options can be presented as either a numerical value or text.

To generate items using models, constraints are also required to eliminate meaningless items. Item models require Boolean logic constraints so that only specific, logically coherent combinations of elements are joined to produce generated items. For example, if two elements, *I2* and *I1,* in an item model cannot be presented with the same value, then the item model is constrained from producing unwanted items by adding logic constraints in the form of *[[I1]] != [[I2]]*. Using common gate logic [e.g., AND (&&) or OR(||)] and Boolean comparison symbols, this type of constraint becomes more complicated if the elements are text (e.g., if the third variation of *S1* cannot appear with second variation of *S2*, then ([[*S1*]] != 3)&&([[*S2*]] != 2)). To ensure models only generate items from desired combination of elements, multiple constraints can also be added to IGOR to limit the generated instances of each item. Constraints are also applied on distracters to minimize the presentation of unwanted combinations of options (i.e., if a pair of options should not be presented with the same outcome). In IGOR, constraints are checked for every generated item to ensure elements in an item are logical and desired combinations. Items that do not meet the demand of a constraint are removed by the program.

Practical multiple-choice item characteristics must also be considered when generating items. These characteristics include the order of item options,

the number of options, presentation of option keys (e.g., A,B,C vs. 1,2,3), and the inclusion of any auxiliary information to the items such as passages, images, tables, or figures. IGOR can accommodate these characteristics during programming. For example, Figure 26 shows the number of ways options can be displayed. Options can be presented with the key as the first option, key as the last option, ascending alphabetical order, descending alphabetical order, or a randomized order. Also, IGOR can be used to generate items having no options (e.g., student-produced response items).

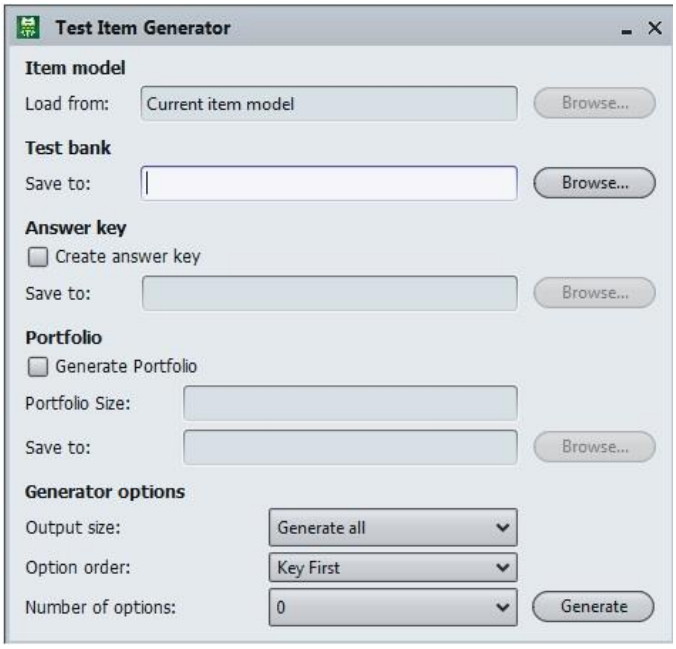

Figure 26. Options to modify item generation characteristics in IGOR.

Generating Items from IGOR

After an item model is programmed, IGOR generates all logical instances of the item model. Also, if the item model contains more options than required

in the item model (e.g., generating items with four options when there are five valid options in the item model), then the same item stem is reused to generate items with different options. In an item model with no constraints, the number of generated items can be calculated by:

$$n = \prod_e c_e,$$

where *e* is the number of elements within the item model, and *c* is the range (i.e., number of variations) for element *e*. This equation demonstrates that the number of variations in each element dramatically increases the number of generated items. Conversely, when constraints are added to the item model, the number of generated items will decrease.

After generation, items are stored in a test bank that can be viewed in any word processor or Internet browser. The bank is organized in an XML format. Figure 27 is a generated item bank displayed in a web browser. An answer key associated with the test bank can also be generated.

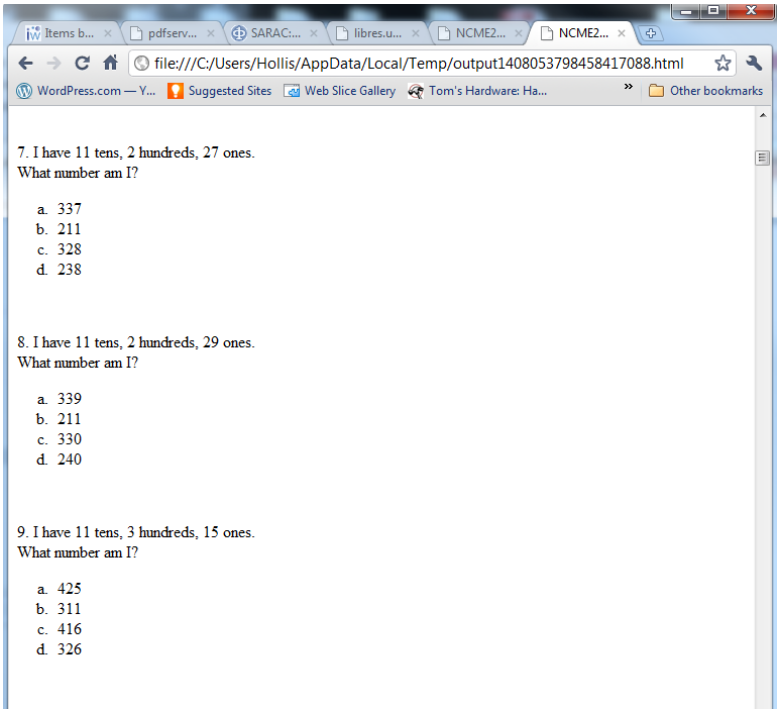

Figure 27. An example of an item bank generated by IGOR.

Hernia

Two item models were created to generate hernia items. Using the 1-layer hernia item model, information is entered through the IGOR user interface. Specifically, the stem, elements, options, and constraints are defined and saved in an extended markup language(XML) format. Other features in IGOR are also defined. For example, the generated options are randomized. Recall that the total number of combinations in item generation is the product of all ranges in each element. In the 1-layer item model with seven elements, a total of 16,384 possible combinations are created by IGOR. But, based on the required constraints of the model, only 256 combinations are permissible, hence, only 256 items are generated. A sample of the generated items is presented in Figure 28.

11. A 35-year-old woman presented with a mass in the left groin. It occurred a few months ago. On examination, the mass is protruding but with no pain and lab work came back with normal results. Which of the following is the next best step?

a. ice applied to mass*
b. exploratory surgery
c. reduction of mass
d. hernia repair

50. A 30-year-old man presented with a mass in an area near a recent surgery. It occurred a few months ago. On examination, the mass is protruding but with no pain and lab work came back with normal results. Which of the following is the next best step?

a. ice applied to mass*
b. exploratory surgery
c. reduction of mass
d. hernia repair

175. A 55-year-old woman presented with a mass and severe pain in the umbilicus. It occurred a few days ago. On examination, the mass is tender and exhibiting redness and lab work came back with elevated white blood cell count. Which of the following is the next best step?

a. ice applied to mass
b. exploratory surgery
c. reduction of mass
d. hernia repair*

137. A 25-year-old woman presented with a mass and severe pain in the left groin. It occurred a few days ago. On examination, the mass is tender and exhibiting redness and lab work came back with elevated white blood cell count. Which of the following is the next best step?

a. ice applied to mass
b. exploratory surgery
c. reduction of mass
d. hernia repair*

*-correct option

Figure 28. Generated items from 1-Layer Hernia item model.

With the n-layer item model in hernia, the stem, elements, options, and constraints are entered into IGOR. This model produced a total of 786,432 combinations, but only generated 12,287 items based on the constraints of the

model. A sample of the generated items from the n-layer model is presented in Figure 29. The XML code for both item models are presented in the Appendix A.

5326. A 50-year-old man presented with a mass and mild pain in the left groin. It occurred a few days ago after moving a piano. Upon further examination, the patient had normal vitals and the mass is tender and reducible. Which one of the following is the best treatment?

a. exploratory surgery
b. reduction of mass
c. hernia repair*
d. ice applied to mass

4610. Patient complaints of a mass in the left groin which has been a problem since a few months ago. On examination, the mass is protruding but with no pain and lab work came back with normal vitals. Which one of the following is the best treatment?

a. reduction of mass
b. exploratory surgery
c. hernia repair
d. ice applied to mass*

12010. Patient complaints of a mass and mild pain in the umbilicus which has been a problem since a few days ago after moving a piano. There is tender and reducible in the umbilicus and the patient had normal vitals. Given this information, what is the best course of action?

a. exploratory surgery
b. ice applied to mass
c. reduction of mass*
d. hernia repair

7325. A 45-year-old man presented with a mass and severe pain in right groin. It occurred a few days ago. There is tender and exhibiting redness in the right groin and the patient had elevated white blood cell count. Which one of the following is the best treatment?

a. reduction of mass
b. hernia repair*
c. ice applied to mass
d. exploratory surgery

*-correct option

Figure 29. Generated items from N-layer Hernia item model.

Post-Operative Fever

With the 1-layer post-operative fever item model, five elements are required for the stem. The options for this model require four variables resulting in a model with 1,399,680 possible combinations. The addition of constraints restricted the model output to 1,248 items. A sample of these generated items are presented in Figure 30.

1. A 34-year-old woman has an appendectomy. On post-operative day 6 he has a temperature of 38.2 C. Physical examination reveal tenderness in the abdominal region with guarding and rebound. Which one of the following is the best next step?

a. Drainage*
b. Reopen the wound
c. Antibiotics
d. Mobilize

2. A 46-year-old man was admitted to the hospital for an appendectomy. On post-operative day 4 he has a temperature of 38.2 C. Physical examination reveal tenderness in the abdominal region with guarding and rebound. Which one of the following is the best next step?

a. Drainage*
b. Anti coagulation
c. Reopen the wound
d. Mobilize

3. A 54-year-old woman has a laparoscopic cholecystectomy. On post-operative day 4 she has a temperature of 38.2 C. Physical examination reveal a red and tender wound at the opening. Which one of the following is the best next step?

a. Reopen the wound*
b. Mobilize
c. Antibiotics
d. Anti coagulation

4. A 62-year-old man was admitted to the hospital for a laparoscopic cholecystectomy. On post-operative day 1 he has a temperature of 38.2 C. Physical examination reveal no other findings. Which one of the following is the best next step?

a. Mobilize*
b. Reopen the wound
c. Antibiotics
d. Drainage

*-correct option

Figure 30. Generated items from 1-Layer Post-Operative Fever item model.

With the n-layer model, two layers of elements are used to manipulate sentence structure and options. From a total of 201,553,920 possible combinations, the n-layer model generated 10,350 post-operative fever items. A

sample of the generated items are presented in Figure 31. The XML code used to generate items for both models are presented in Appendix B.

1415. A 43-year-old woman has a laparoscopic cholecystectomy. On post-operative day 4 she has a temperature of 38.5 C. Physical examination reveal a red and tender wound at the opening. Which one of the following is the most likely diagnosis for this patient?

a. Atelectasis.
b. Wound infection.*
c. Deep space infection.
d. Urinary tract infection.

4877. A 47-year-old patient who had a left hemicholectomy had a fever of 38.5 C on post-operative day 6. Physical examination reveal tenderness in the abdominal region with guarding and rebound. Which one of the following is the most likely diagnosis for this patient?

a. Pneumonia.
b. Wound infection.
c. Deep space infection.*
d. Urinary tract infection.

8594. A 55-year-old woman has a gastrectomy. On post-operative day 3 she has a temperature of 38.5 C. Which one of the following is the most likely diagnosis for this patient?

a. Wound infection.
b. Atelectasis.
c. Deep vein thrombosis.
d. Urinary tract infection.*

9218. A 55-year-old patient who had a laparoscopic cholecystectomy had a fever of 38.5 C on post-operative day 3. Which one of the following is the best next step for this patient?

a. Mobilize.
b. Prescribe antibiotics.
c. Reopen wound.
d. Prescribe anti-coagulation medicine.*

Figure 31. Generated items from the n-layer post-operative fever item model.

Summary

In this chapter, I applied the SIG framework to item generation in the context of medical education. I reviewed the Qualifying Examination Part 1 of the Medical Council of Canada (MCCQE-1) and I provided some cognitive modeling concepts from the field of medical education that were used for item generation. Then, I presented an application of the three stages of SIG for generating items in hernia and post-operative fever. To model the content expert knowledge, I used a cognitive model knowledge structure for item generation that captured relevant information from the content experts. I also introduced the use of n-layer item models to vary the text similarity of the generated items. In the next chapter, I describe methods that can be used to evaluate the text similarity among the generated items.

Chapter V: Measuring Similarity Among Generated Items

The purpose of automatic item generation is to produce large sets of new test items. The SIG framework demonstrated how items, designed to measure a set of scenarios related to a common problem, can be generated using a systematic process. These new items populate item banks which, in turn, lowers the exposure rate of each item in the bank. Lowering exposure reduces item theft and helps ensure the testing process is fair for all examinees. While a common step in development after generation is to estimate the psychometric properties of the items, another required task is to evaluate the similarity of the generated items. If the goal of item generation is to minimize exposure, then the generated items should not be overtly similar to one another. To help address the problem of homogeneity among the generated items, I developed n-layer item models. N-layer modeling yields generated items that are heterogeneous. However, no quantitative method is currently available for evaluating the similarity of the generated items. In this chapter, I present two new methods for comparing and evaluating the similarity of generated items. I first summarize the current approaches used to evaluate text similarity. One of the methods, called the cosine similarity index (CSI), is used to compare generated item similarity in this study. Then, using the CSI, I present two levels of comparison, intra-model and inter-model differences, for evaluating the similarity of the generated items. Finally, I apply these methods to the generated items presented in the previous chapter.

Evaluating Text Similarity of Generated Items

With the potential of generating thousands of items from a single item model, no researcher has yet attempted to quantify the text similarity of generated items. To evaluate item similarity, a comparison of all unique item pairs within a set of items must be computed. A holistic evaluation by raters can be used to judge the similarity of different test items (e.g., ask content specialists whether a set of items look similar to one another). However, this approach quickly becomes ineffective because the number of required comparisons increases exponentially. For example, pairwise comparisons for a set of 10 items require $\binom{10}{2}$ = 45 unique comparisons, whereas pairwise comparisons between 11 items require $\binom{11}{2} =$ 55 comparisons. The exponential increase of pairwise comparisons render human ratings infeasible, particularly given that item generation has the ability to produce thousands of items from a single item model. As result, an alternative method for systematically evaluating the text similarity of generated items is needed.

Advances from natural language processing have offered some innovative quantitative methods for evaluating text similarity. Currently, three general approaches exist for measuring the similarity of texts. The three methods are n-gram similarity, semantic distances, and cosine similarity. The three approaches vary in their computational complexity. While all three

methods evaluate similarity based on pairwise comparisons between two strings of text, each method is suitable for comparing different types of text.

The first approach is to count the number of overlapping words between two strings. This approach, named n-gram similarity, is the proportion of overlapping words (i.e., grams) counted in different consecutive lengths (i.e., n-length). Take, for example, the following two sentences:

a) What is your name?

b) What is your favourite pet?

If the two sentences are compared using bi-grams, meaning texts with consecutive word length of 2, then the similarity between the two sentences is one (e.g., What is) out of seven unique bigrams. This n-gram approach relies on consecutive word overlap between two sentences. However, some researchers have suggested that n-gram comparisons should only be used for long lengths of text because the co-occurrence of common words (i.e., stop words) in short texts would artificially inflate similarity (Lin, 2004).

The second approach for evaluating text similarity is semantic distances. This complex approach measures the relatedness of words from one text to the other. For example, consider the following three words: cat, dog, mock. The semantic distance between cat-dog is relatively closer than cat-mock and dog-mock, as both cat and dog are related to each other in many other words (e.g., animal, pet). Although semantic distances is a powerful method to quantify and

measure text similarity, it requires a word net or common corpus that have established associations plotted between large sets of commonly used words (see Belov & Knezevich, 2008). Because these associations are not readily available for different content areas in item generation (e.g., medical), the use of semantic distances requires substantial background research to establish relationships within a common corpus and, hence, is not suitable for the medical examples used in my dissertation research.

The third method for measuring text similarity is called the cosine similarity index (CSI). The CSI is a measure of similarity between two vectors of co-occurring texts, where the overall comparison is made by measuring the cosine of the angle between the two vectors in a multidimensional space of unique words. The CSI can be expressed as

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|},$$

where A and B are two items expressed in a numerical vector of word occurrences. For example, if A is a list of three words (dog, walk, talk) and B is a list of another (cat, walk, mock), then the length of both binary vectors is equal to the number of unique words used across both lists (dog, walk, talk , cat, mock). To vectorize sentences A and B for comparing text differences, the occurrence of each word in the corresponding list is quantified with a value of 1. The resulting word occurrence vectors for A and B in this example is [1,1,1,0,0]

and [0,1,0,1,1], respectively. With the dot product between vectors A and B being 1, and the vector length for both A and B being the square root of 3, the inner product of these two vectors result in a CSI of $\frac{1}{3}$ ~ 0.33. The CSI has a minimum value of 0, meaning no word overlapped between the two vectors and a maximum of 1, meaning the text represented by the two vectors are identical. Compared to n-gram approaches, the CSI is more sensitive to shorter lengths of text as it interprets co-occurring words into multidimensional space. Compared to semantic distances, the CSI is a simplified approach of text comparison that does not account for word relatedness in calculating similarity. With the only requirement being a list of common words between the compared texts, the CSI is suitable and generalizable for evaluating text similarity with different item types and also any number of generated items.

Two Levels of Evaluating Item Similarity

To evaluate item similarity, some operational definitions are needed before establishing this method of comparison. First, a text similarity index such as the CSI is required to measure the similarity from one item to another. However, many pairwise comparisons are required when evaluating similarity among a large set of generated items. Recall that the number of pairwise comparisons increases exponentially as the number of generated items increases. Therefore, a new scale of analysis is required to summarize text similarity in an omnibus manner. Second, as items are generated from models, the variation among the generated items is systematically manipulated by the

features within the item model. Therefore, similarity among the items generated from the same model should be related. Given these two demands in evaluating item similarity, I propose two levels of evaluation for summarizing the CSI of generated items, intra-model and inter-model differences. These two levels of analysis are designed to provide different types of information about the similarity among the generated items.

Intra-Model Differences

The intra-model difference is the first level of analysis used to describe text similarity. By calculating the differences in the text for all unique item pairs using the same model, the mean of these CSIs provide an omnibus measure of text similarity. Moreover, because the text of each generated item is systematically manipulated by the item model, the standard deviation of these CSIs can also help describe the variability of the text between different generated item pairs. Because items are generated using systematic feature manipulations, a random sample of generated items can be used to calculate the intra-model differences for all the generated items. A random sample of items is needed for practical reasons because pairwise comparisons increase exponentially as the number of generated items increases thereby causing memory problems for computation and as the number of generated items for each model differs, a common number of items is needed to compare between different item models. In sum, intra-model differences can be summarized by the mean and standard deviation of the CSI from a random sample of pairwise-

comparisons using items generated from the same model. A high CSI mean represents an item model that generates highly similar items (i.e., homogeneous), whereas a low CSI mean represents an item model that generates items with low similarity (i.e., heterogeneous). A low CSI standard deviation represent low variability in the generated item differences (i.e., homogeneity), whereas a high CSI standard deviation for a given model represent high variability in the generated item differences (i.e., heterogeneity).

Inter-Model Differences

After intra-model differences are computed, a second level of analysis can be conducted to compare the differences between item models. Comparison of differences between models are computed based on the intra-model differences (i.e., models are independent), rather than calculating a new set of CSI between items of different models (i.e., models are dependent) because the unit of analysis is text similarity between item model. In other words, text differences are summarized as a distribution of text differences for each item model and these differences are then compared between item models. In sum, information on inter-model differences can be used to evaluate whether the text similarity of an item model is significantly varied during generation or to provide evidence for evaluating whether different item models can generate items with comparable levels of variability. Next, I demonstrate how these measures can be applied to the generated items from the previous chapter so conclusion about text similarity can be made. Also, I evaluate

whether the use of n-layer item models reduced similarity among the generated items.

Comparison of Generated Items

The purpose of n-layer item modeling was to provide an alternative to the current 1-layer item modeling approach as a way to increase variation among the generated items. The benefits of n-layer modeling are obvious when it comes to increasing the generative capacity of the item model. Compared to traditional 1-layer modeling approach, the n-layer item model generated 48 times more items in hernia (12,287 compared to 256) and six times more items in post-operative fever (10,350 compared to 1,504). However, to evaluate whether n-layer model provide more text variation among the generated items, comparisons of text similarity are needed. To begin, I outline the procedures necessary for cleaning and organizing the generated items for analysis. Then, I summarize the intra-model differences of each item model. Finally, I statistically evaluate the inter-model differences using the same content but generated using the 1-layer and n-layer item model.

Procedure

To evaluate item similarity from any given model, a random-sample of 100 generated items was selected from the entire generated item set. This sample of items is then used to calculate a total of 4,950 unique pairwise comparisons. The CSI value is calculated for each item-pair. Using the statistical

programming language R (R Core Development Team, 2012), the following process was used to calculate the CSI between each pair of items. First, the stem and options of each item were concatenated and formatted into a string of lower-cased words. All characters in this analysis need to be in a lower-case form because the comparison process and identification of unique words are case-sensitive. Second, common words (e.g., a, is, was), special characters (e.g., %, &, *), and punctuations are removed from the string. Common words were removed based on a list of the 300 most commonly used words in English from the LSA program in R (Wild, 2011). The removal of common words is a popular method in NLP to minimize the artificial inflation of similarity caused by words that are irrelevant to the context of the strings. Special characters and punctuations also muddle the identification of unique words. For example, if a word appended with a punctuation at the end of a sentence is not adjusted (e.g., fever.), then the word "fever." would be recognized as a unique word from the original word "fever". Such errors would then artificially deflate the level of similarity in each item pair. After adjustments are made to the words, a list of unique words from all generated items is compiled. This process is completed using programming, where unique words are iteratively identified for all generated items and are then recorded into a list. The unique word list is used to populate a word-occurrence matrix. The word-occurrence matrix records the corresponding use of a given word for each string, where each item is represented as a row, each column represents a unique word, and the cell value

of the corresponding row and column is the appearance of the word on a given item. In this study, I chose the use of binary vectors, meaning I only record whether or not a word was used within a given item. The use of binary vectors in the word-occurrence matrix helps minimize the artificial inflation of similarity given to items that use the same term multiple times. For example, if the word "patient" is used multiple times, then recording the count would inflate the level of similarity between item pairs. Finally, to calculate similarity between an item pair, each row of the word-occurrence matrix becomes the vector used to calculate the CSI, where programming is again used to systematically calculate the CSI for all unique item pairs.

Intra-Model Differences

The intra-model differences for the four item models are presented in Table 6. For the 1-layer item models, the CSI for hernia ranged from 0.55 to 0.98 with an overall mean of 0.74 and a standard deviation of 0.11. The CSI for post-operative fever ranged from 0.54 to 1.00 with an overall mean of 0.77 and a standard deviation of 0.08. The mean CSI for an item model represents the central tendency of the intra-model differences for all item pairs within the model. Recall that a CSI of 1 represents an identical item pair while a value of 0 represents a completely unique item pair. Hence, the means of 0.74 and 0.77 suggest a relatively similar level of text similarity for both 1-layer surgery item models. The standard deviations of 0.11 and 0.08 indicate that the similarity among the items within each model is high.

Table 6.

Summary of Cosine Similarity Index as a Function of Content Area and Model Type

	Min	Max	Mean	SD
Post Op Fever				
1-layer	0.54	1.00	0.77	0.08
N-layer	0.13	1.00	0.51	0.19
Hernia				
1-layer	0.55	0.98	0.74	0.11
N-layer	0.17	1.00	0.53	0.16

For the n-layer item models, CSI for hernia ranged from 0.17 to 1.00 with an overall mean of 0.53 and a standard deviation of 0.16. For post-operative fever, the generated items produced CSI values ranging from 0.13 to 1.00 with a mean of 0.51 and a standard deviation of 0.19. Relative to the results from the 1-layer item models, it appears that adding a layer of elements did reduce text similarity between generated items. The means of 0.53 and 0.51 suggest a relatively dissimilar level of text similarity relative to the 1-layer surgery item models. The standard deviations of 0.16 and 0.19 indicate that the similarity among the items within each model is low relative to the 1-layer surgery models.

Inter-Model Differences

The inter-model differences can be illustrated with a histogram of the CSI values. Figure 32 presents the distribution of CSI values for the 1-layer and n-layer model for hernia. From this graph, it is apparent that the n-layer item model increased text differences within the model. Moreover, the spread of the CSI values increased between the 1-layer and n-layer models. Results from Figure 33 demonstrate a comparable pattern of results when the similarities of 1- and n-layer post-operative fever models are plotted.

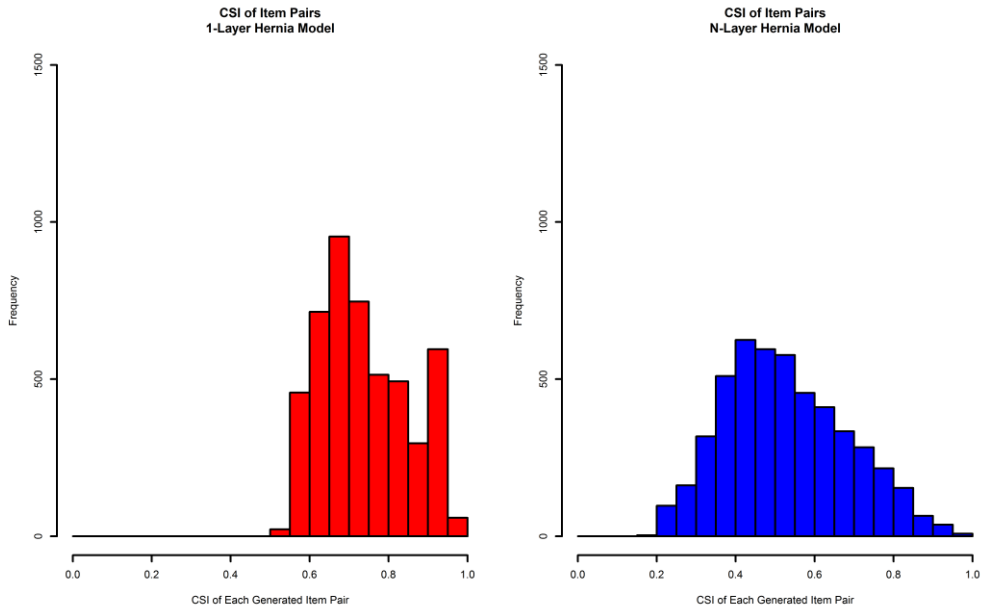

Figure 32. Histograms of the CSI values for each item pair for Hernia item models.

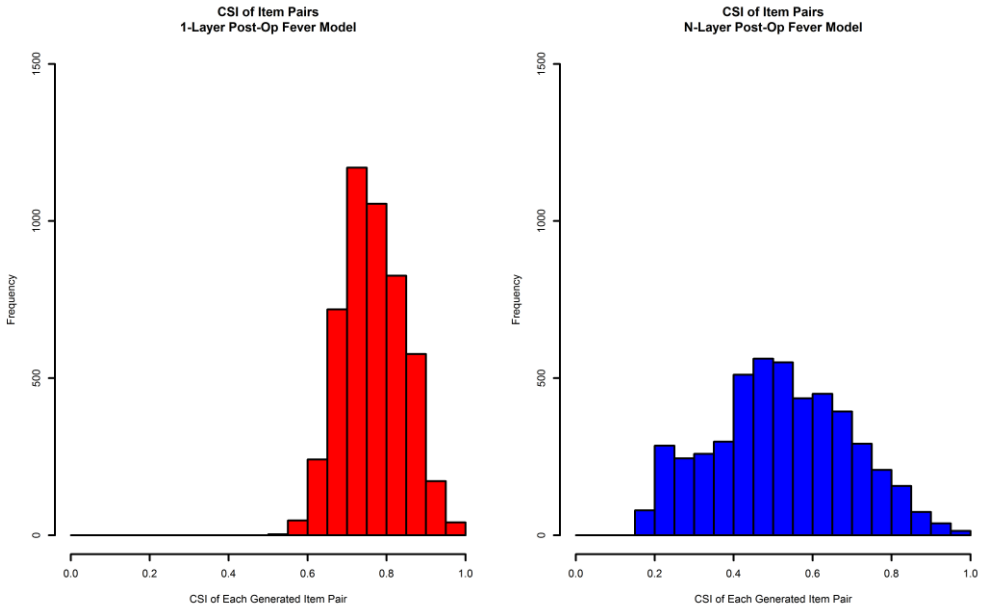

Figure 33. Histograms of the CSI values for each item pair for the Post-Operative Fever item models.

To statistically evaluate the trends illustrated in the histograms, independent samples t-tests were conducted to determine whether the use of n-layer item model significantly reduces text similarity relative to the 1-layer item model. The t-tests used the intra-model differences, or CSI values of item pairs within its own respective model, as the dependent variable, and the item model type as the independent variable (1-layer or n-layer). For items generated from the hernia models, the *t*-test revealed a statistically significant difference between the item model types, $t(9899) = 77.18$, $p < 0.05$. A significant difference was also found on post-operative items between item model types, $t(9899) = 84.12$, $p < 0.05$. Taken together, these results provide evidence that items generated from the n-layer item models do, in fact, produce more

heterogeneous and diverse items compared to those generated from 1-layer item models.

Summary

In this chapter, I introduced a method to evaluate item similarity and I demonstrated how this method can be used to compare the items generated using the 1-layer and n-layer models. Results from my demonstration reveal that not only do n-layer item models generate more items, but they also create more heterogeneous items compared to traditional 1-layer models. The methods presented in this chapter are beneficial for evaluating the generative capacity of an item model and for providing an alternative measure to evaluate the text similarity of items produced from different AIG modeling methods.

Chapter VI: Discussion

Four years ago, I described an item generation process to an audience of educational experts at an academic conference (Lai, Alves, Zhou & Gierl, 2009). The audience members and session discussant viewed the idea as futuristic, but with little practical implication. Admittedly, the item generation process at the time was cumbersome and complex, test developers were overwhelmed by the requirements of the task, and the generated items were criticized because they appeared too similar to one another. A considerable amount of research on item generation has taken place since 2009. Cognitive scientists have conceptualized and improved on how information can be modelled for item generation. Computer scientists provided technological innovations on the generative process itself. Psychometricians proposed different statistical modeling methods to estimate the psychometric properties of the generated items. Together, these developments describe a new state-of-the-art for item generation that requires an application framework. The purpose of my dissertation was to create an application framework and summarize the different stages of development required for generating items to demonstrate how a systematic process could be used to produce new test items. To reduce the level of complexity required in the generation process, items can now be produced using the systematic development process describe in my dissertation research. Test developers, who are essential for the success of AIG, are now charged with the task of designing and developing meaningful item models as

well as identifying the required content. Computer technology, which is also essential for the success of AIG, is charged with combining large amounts of information in a systematic manner required to produce new test items. By combining content expertise with computer technology, models can be created that yield large numbers of high-quality items in a short period of time. In this chapter, I present a summary of the methods introduced in my study. I discuss the potential limitations to my approach. I also provide an outlook on future developments required for item generation.

Summary of Study

For my dissertation research, I presented a systematic item generation framework (SIG) to produce medical test items. The SIG framework consisted of three development stages to facilitate the modeling of content expert knowledge into cognitive models, the translation of cognitive models into prototypes for generation, and the generation of test items using computer-based algorithms. This framework is used to guide and support a manufacturing process to produce large numbers of test items. Using this framework, I generated 22,637 test items in the content areas of hernia and post-operative fever for a medical licensure testing program. The implications of these methods and their contributions to the field of item generation are discussed next.

Modeling Knowledge for Item Generation

Content experts rely on guidelines to produce items from a test specification or content domain blueprint. In item generation, an intermediate modeling process is needed for translating item specifications into content that can be used to produce new items, as generating large numbers of test items require content experts to express their knowledge for multiple items collectively. In my dissertation, I presented a cognitive model structure to capture information from content experts to identify the content, knowledge, and skills needed for item generation. The cognitive model structure can be broken down into three components of content. *Problems and scenarios* provide the issues, both general and specific, probed across an entire set of generated items. *Sources of information* organize all cues and features for the generated items. *Features* are the specific characteristics associated with each information source. In any cognitive model, multiple features may be presented under a single source of information while the presentation of features vary for each generated item. In my study, I demonstrated this cognitive modeling process by collecting the required information for generating test items in hernia and post-operative fever using a semi-structured interview format. The outcome of this process produced a cognitive model for generating items related to managing four different hernia issues under the expression of ten features. It also produced a cognitive model for generating items in six different scenarios under post-operative fever using 12 unique features. In short, the process I

demonstrated in my study provides a description of how content expert knowledge can be captured for item generation. Haladyna and Gierl (2012) predicted that the development of a knowledge structure to support item generation would become the main obstacle in item generation research. The contribution of this study is to outline the required processes for developing such knowledge structures to enable future developments in item generation.

Item Modeling and Generation

After knowledge has been extracted from content experts, I described and demonstrated item modeling, a process to structure content expert knowledge into a format that allows for item generation using a template-based automatic item generation (T-AIG) approach. In contrast to a cognitive model, which captures a transfer of knowledge from content experts, item models incorporate information from the cognitive model into a modular format suitable for creating new test items. An item model is therefore a prototypical representation of the set of test items to be generated. Because current item modeling techniques are often criticised for producing items that appear too similar to one another, I also introduced and demonstrated the use of a modeling technique called n-layer that allows elements to be embedded within one another, thereby permitting larger variations between the generated items. In my dissertation, I demonstrated the production of four item models using 1-layer and n-layer modeling techniques in both content areas. Then, using the item generation software IGOR, I demonstrated the third and remaining stage in

the SIG framework and described how item models along with binary programming constraints can be used to generate desirable items that fit with the demands of the cognitive model. The contribution of my study is to provide a comprehensive demonstration on the item modeling and generation processes required to produce large numbers of test items from content expert knowledge. Moreover, my dissertation results demonstrate how n-layer item models can increase both the number of generated items and the variability between generated items.

Evaluating Item Similarity

After items are generated, the next stage of development often involves field testing and evaluating the psychometric properties of the generated items. This psychometric analysis is problematic when thousands of test items are produced through the AIG process. Moreover, the psychometric analysis provides the developer with information about the statistical characteristics of the items but no information is available about the similarity of the generated items. In my dissertation, I proposed and demonstrated a method for evaluating text similarity among the generated items from the same item model. The Cosine Similarity Index(CSI) is a measure of the similarity between text for the generated items. To compare text similarity between items, word occurrences of each item are vectorized into a set of unique words common across all items. To analyze these differences, two approaches can be used. By systematically comparing a random set of generated items in a pairwise manner, the resulting

comparisons across all items within a model provides a description of intra-model differences, or variability of items within the model. By conducting significance tests between results from different models, a summary of inter-model differences, text variability can be compared between models. These methods of comparisons were used to determine the similarity of each item model thereby providing evidence that n-layer models generate items with less text similarity.

Limitations

No Evaluation or Calibration of Generated Items

The purpose of my study was to demonstrate a process to generate test items. Ideally, my study would have included field testing results to evaluate the quality of the generated items and provided estimates of their psychometric properties. Item quality can be evaluated using judgments from content specialists where the guidelines, conventions, and standards of practice for creating multiple-choice items form the basis of scrutinizing the items. Item quality can also be evaluated using a psychometric approach where the items are pilot tested to yield information about their statistical characteristics (e.g., *p*-values). The availability of field testing results would have also enabled the demonstration of related sibling family method for calibrating generated items. Unfortunately, the collection of field testing results required substantial resources beyond the scope of my dissertation. Lai & Gierl (2012) outlined some

design features that should be considered when generated items require statistical calibration but, to-date, the calibration of generated items using SIG framework has not occurred. As a result, a traditional investigation on generated item quality and a summary for estimating the psychometric properties of generated items was beyond the scope of this study. Despite the apparent feasibility and potential usefulness of AIG as demonstrated in this study, future studies need to collect field test results on the generated items to determine the quality of these generated items relative to those items developed using a more traditional test development approach. Moreover, a general framework for estimating the psychometric properties of the generated items is needed to complete the development process for generated items that allows for their operational use.

Knowledge Modeling Reliant on Content Experts

Item writing relies on the knowledge and experience of the content experts. Guidelines provide best practices, common mistakes, and general expectations of item writing to ensure content experts have a shared understanding of the task at-hand so high-quality test items can be produced (Haladyna, 1994; Case & Swanson, 1998). Current item generation approaches assume that content experts are able to express their knowledge in the format of the cognitive model. This assumption is also a limitation of my study and of the current approaches to AIG, more generally. Content experts are assumed to be able to express their knowledge in a form that permits item generation. But

more training may be required to extract their knowledge in a manner that is captured in the cognitive model. In other words, while item writing is inherently an exercise of deduction (e.g., finding a precise combination of elements and expressing them in the form of a test item), content experts may not be adequately expressing the same set of required knowledge inductively for generation (e.g., defining all possible features given a specific set of diagnoses). Although the cognitive modeling methods have been replicated in other areas (Gierl, Lai, & Breithaupt, 2012), more evidence is needed to ensure content experts are able to elicit their knowledge inductively and provide knowledge in the proposed cognitive modeling approach. In our experience, some content specialists are better than others at this inductive reasoning process. However, we don't know why these skills vary across content specialists or how to more effectively teach specialists to use these skills during the cognitive model development stage.

Generation of Distracters

The item generation process described in my dissertation explicitly models for the stem and correct option of each generated item, but it only implicitly models for the corresponding incorrect options or distractors. However, generating multiple-choice questions require items to not only include a stem with a corresponding correct response but also a set of options with the corresponding incorrect responses. Hence, a limitation of this study is a lack of methodology for generating distracters. Incorrect options for multiple-choice

items should be “plausible, grammatically consistent, logically compatible, and of the same (relative) length as the correct answer” (Case & Swanson, 2001). In the context of item generation, incorrect options should be designed from a list of plausible but incorrect alternatives linked to common misconceptions or errors in thinking and reasoning. In sum, when items are generated, a parallel generation process is needed to systematically produce the incorrect options for each generated item. Some test development practices require content specialists to provide algorithms or sets of rules based on errors or misconceptions that can be used to produce distractors. This type of practice allows for *systematic distractor generation*, which has been demonstrated in a small number of quantitatively-based content areas such as mathematics (e.g., Lai, Gierl, Alves, 2010). But for domains such as medicine and for skills such as clinical reasoning and medical problem solving, simple rules or well-established algorithms are rarely available to model errors in thinking, reasoning, and problem solving needed to support systematic distractor generation. In my dissertation, I compiled a list of distracters by presenting all treatment or management options associated with the scenarios within the cognitive model. With this approach, I was able to generate distractors that were related (e.g., through the problem in the cognitive model) but still different from the correct option. Clearly, however, this approach is not sufficient for providing options as the availability of distracter, and the relatedness between distracters, are dependent on the structure of the cognitive model. Hence, a more systematic

approach for producing distracters is needed. Currently, no generalized approach for developing multiple-choice distracters exists. Future research must therefore be undertaken to develop and evaluate a systematic approach for generating distracters which, in turn, can be incorporated into existing item generation frameworks.

Future Directions

Item generation is still considered to be a young and promising topic of research despite developments in the past 50 years (Haladyna & Gierl, 2012). Researchers across different disciplines (e.g., computer science, psychometrics, and psychology) have all contributed specific solutions tailored specifically to meet the needs in each field. However, research on item generation has also been segmented (Haladyna & Gierl, 2012), yielding many attempts, but little impact, to address the demand for new items and correct the deficiencies in our current item writing practices. The goal of my dissertation was to develop and demonstrate a framework of item generation that *componentializes* different stages of AIG so test developers can implement the framework. As the demand for item generation solutions become more popular, I foresee three future areas of research needed in this field.

Establishing a Science for Item Generation

Haladyna (2012, p. 19) noted that "given the gaps in theory and research on item generation and the slow and uneven development of AIG dating back to

the early 1960s, we have had periodic introductions of prescriptive methods for item generation. These methods have no strong link to any theory but may resemble or capitalize on implied technologies from these past attempts at item generation". That is, Haladyna highlighted the vulnerability of item generation development to existing technologies. He also noted that few theories are available to link developments between new AIG efforts. Recognizing that sustained and continual efforts in item generation are needed to ensure its success, a science or organized form of development on a body of agreed upon knowledge is required. In other words, a science of item generation is needed. This science will promote continual innovations and allow researchers to collaborate and refine a unified methodology that is robust to the constant change in technology.

However, it should be noted that addressing the need for a science of item generation differs from the assumption that items can be scientifically generated. Although item generation has often been thought of as a replacement of human involvement in the item development process, established roles of both technology and content experts would suggest that both are needed. Item generation can be viewed as a hybrid approach that requires both technological sophistication and content expertise (Gierl & Lai, 2011). This collaborative approach requires the artistic expression of content expert knowledge in the form of cognitive models with the scientific expression of raw computing power to facilitate an iterative process of assembling large

numbers of items. Simply put, item generation requires both the art and science in test development to produce large numbers of items. Moreover, as a science of item generation begins to develop, researchers can begin to solve issues beyond a scope not limited to a specific, hence technologically dependent, generation approach thereby addressing Haladyna's (2012) recent concern.

Generating Different Item Types

The multiple-choice format is an essential item type for developing medical examinations across different content areas and scales of administration. In my demonstration of the item generation framework, multiple-choice questions were generated because this format is used on the MCCQE-I. It is also the item type most commonly administered in medical education in 2012. But as computer-based testing becomes more pervasive, different item types will appear and begin to influence our item development practices. For example, Scalise and Gifford (2006) presented a taxonomy of 28 innovative item types organized by two dimensions, seven levels student involvement was crossed with four levels of response complexity. As item generation begins to produce test items, methods are needed to generate different item types. Currently, generation methods are unique to each item type, where few studies have presented unique methods to generate probability word problems (Holling, Bertling, & Zeuch, 2009), figural reasoning, where unique images are generated for each item for examinees to match (Arendasy, 2010), in addition to multiple-choice items. Because the framework presented in

my dissertation is not limited to a single item type, future studies can attempt to modify the framework by adjusting the generation technology to produce different item types. With a modular approach to the entire item generation process, the framework presented in my dissertation can also serve as the initial basis for enabling such development.

Item Model Characteristics

As item generation becomes capable of producing large numbers of test items, more information is needed to describe the generated items. Currently, psychometric methods are available for describing statistical information at the item level, but as mentioned in Chapter 2, such information is troublesome to produce when large numbers of generated items are available. Examples of estimating and summarizing item model properties do not yet exist. Therefore, methods are needed to describe the characteristics of the generated items at the item model level to allow effective administration of the items as an instance of the item model. Further, item model characteristics that go beyond providing psychometric information can also allow item models to be used for continuous testing. One example of such information is demonstrated in Chapter 5, where text similarity information was summarized between the generated items and presented at the item model level. Information on text similarity can be used to avoid the administration of enemy items or items with similar outcomes. As item models created from my framework are expected to generate items that probe a set of scenarios related to the same problem, future studies can develop

methods to map the content required to enable systematic administration of generated items. In sum, more information is needed at the item model level if generation is to be used for testing.

Conclusion

Technology is changing the way students interact and learn. As a result, educational testing is also undergoing dramatic changes. Testing is no longer a rite of passage. Rather, tests can be administered online, regardless of location, at any time. Testing is no longer a summation of the learning process. Rather, testing is used throughout educational process to initiate, refine, and encourage learning specific to each student's needs. These new demands for testing will also affect how we develop test items. In the future, an unprecedented need for testing items, measuring more specific skills and providing more information about student learning, will arise. Item generation is an emerging methodology for developing large number of items in an efficient and cost effective manner. I presented and demonstrated a systematic process for generating items that begins with content expert knowledge and ends with the creation of new test items. As aspects of test development become well integrated with technology, the addition of a systematic generation process to the humanistic task of providing content expert knowledge allow items to be created en masse to meet assessment demands of the 21st century.

References

Alves, C., Gierl, M., & Lai, H. (April, 2010). Using Automated Item Generation to Promote Principled Test Design and Development. Paper presented at the American Educational Research Association. Denver, CO.

Arendasy, M., Sommer, M., & Gittler, G. (2010). Combining automatic item generation and experimental designs to investigate the contribution of cognitive components to the gender difference in mental rotation. *Intelligence,* 38, 506-512.

Arendasy, M., Sommer, M., & Mayr, F. (2012). Using automatic item generation to simultaneously construct german and English versions of a word fluency test. *Journal of Cross-Cultural Psychology, 43,* 464-479.

Association of Faculties of Medicine of Canada (2009). *The Future of Medical Education in Canada: A Collective Vision for MD Education.* Ottawa,ON: AFMC.

Baranowski, R. (2006). Item Editing and Editorial Review. In S. Downing & T. Haladyna (Eds.), Handbook of Test Development. Mahwah, NJ: Lawrence Erlbaum.

Behrens, J., Mislevy, R., Bauer, M., Williamson, D., & Levy, R. (2004). Introduction to Evidence Centered Design and Lessons Learned From Its

Aplication in a Global E-learning Program. The International Journal of Testing, 4, 295-301.

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

Bejar, I. (2002). Generative testing: From conception to implementation. In S. H. Irvine & P. C. Kyllonen (Eds.), Item generation for test development (p.199-217). Hillsdale, NJ: Erlbaum.

Bejar, I. (2010). Model-based item generation: a review of recent research. In S. E. Embretson (Ed.), Measuring psychological constructs: Advances in model-based approaches. Washington D. C.: American Psychological Association Books.

Bejar, I. (2012). Item generation: Implications for a validity argument. In M. Gierl & T. Haladyna (Eds.) *Automatic item generation: Theory and practice.* New York: Routledge.

Bejar, I., Lawless, R., Morley, M., Wagner, M., Bennett, & Revuelta, J. (2003). A feasibility study of on-the-fly item generation in adaptive testing. Journal of Technology, Learning, and Assessment, 2(3). Available from http://www.jtla.org.

Bejar, I., Lawless, R., Morley, M., Wagner, M., Bennett, R. & Revuelta, J., (2002). A feasibility study of on the fly item generation in adaptive testing. GRE Board Report, 98-12. Princeton, NJ: ETS.

Belov, D., & Knezevich, L. (October, 2008*). Automatic prediction of item difficulty based on semantic measures.* Research Report, 08-04. Law School Admissions Council, Newtown, PA.

Bennett, R. (2001). How the internet will help large-scale assessment reinvent itself. *Educational Policy Analysis Archives, 9,* 1-23.

Bormuth, J. (1970). On a theory of achievement test items. Chicago: University of Chicago Press.

Bosma, W., Marsi, E., Krahmer, E., & Theune, M. (2011). Text-to-text generation for question answering. In A. van den Bosch & G. Bouma (Eds.), Interactive Multi-modal Question-Answering, Berlin, Heidelberg, 2011 (p. 117-145). Springer.

Breithaupt, K., Ariel, A., & Hare, D. (2010). Assembling an Inventory of Multistatge Adaptive Testing Systems. In W. van der Linden & C. Glas (Eds.), Elements of Adaptive Testing (p. 247-266), New York, NY: Springer.

Camara, W. (2011). The role of NCME and Measurement Professionals: What Role We May Choose in Policy and Accountability. NCME Newsletter, 19(1), March.

Case, S., & Swanson, D. (2001). *Constructing written test questions for the basic and clinical sciences*. Philadelphia, PA: National Board of Medical Examiners.

Coderre, S., Mandin, H., Harasym, P., & Fick, G. (2003). Diagnostic reasoning strategies and diagnostic success. *Medical Education,* 37, 695-703.

Custers, E., Stuyt, P., & De Vries, P. (2000) Clinical Problem Analysis (CPA): A Systematic Aproach to Teaching Complex Medical Problem Solving, Academic *Medicine.* 75, 291-297.

De Boeck, P. (2008) Random item IRT models. *Psychometrika,*73, 533-559.

Deane, P., & Sheehan, K. (2003). Automatic item generation via frame semantics: Natural language generation of math word problems. Princeton: Educational Testing Service.

Deetmer, K., Krahmer, E., & Theune, M. (2005). Real vs. template-based natural language generation: a false oposition? *Computational Linguistics,* 31 (1).

Downing, S. M., & Haladyna, T. M. (2006). Handbook of test development. Mahwah, NJ: Erlbaum.

Drasgow, F. (2002). The work ahead: a psychometric infrastructure for computerized adaptive tests. In C. Mills, M. Potenza, J. Fremer & W. Ward (Eds.) Computer-based testing: Building the foundation for future assessments. Mahwah, NJ: LEA.

Drasgow, F., Luecht, R., & Bennett, R. (2006). Technology and testing. In R. L. Brennan (Ed.), Educational measurement (4th ed., p. 471-516). Washington, DC: American Council on Education.

Ebel, R. (1951). Writing the test item. In E. F. Lindquist (Ed.), *Educational Measurement* (1st ed., p. 185–249). Washington, DC: American Council on Education.

Elstein, A., Shulman, L. & Sprafka, S. (1990). Medical problem solving: a ten-year retrospective. *Evaluation and the Health Profession*, 13(1), 5-36.

Embretson, S. (1998). A cognitive design system approach to generating valid tests: Application to abstract reasoning. Psychological Methods,3, 300-396.

Embretson, S. (2002). Generating abstract reasoning items with cognitive theory. In S. H. Irvine & P. C. Kyllonen (Eds.), Item generation for test development (p. 219-250). Mahwah, NJ: Erlbaum.

Embretson, S. (2010). Cognitive Design Systems: A Structiral Modeling Approach Applied to Developing a Spatial Ability Test. In S. Embretson (Ed.) Measuring Psychological Constructs : Advances in Model-Based Approaches (p.247-273). Washington, DC: APA.

Embretson, S. , & Daniel, R. (2008). Understanding and quantifying cognitive complexity level in mathematical problem solving items. Psychological Science Quarterly, 50, 328-344.

Embretson, S. E., & Yang, X. (2007). Automatic item generation and cognitive psychology. In C. R. Rao & S. Sinharay (Eds.) *Handbook of Statistics: Psychometrics, Volume 26* (p. 747-768). North Holland, UK: Elsevier.

Ferrara, S. & DeMauro, G. (2006). Standardized Assessment of Individual Achievement in K-12 in Brennan, R. (Ed.) Educational Measurement, 4th Ed. Wesport,CT:NCME/ACE.

Fischer, G. (1973). The linear logistic test model as an instrument in educational research. Actas Psychologica, 37, 359-374.

Flaugher, R. (1999). Item Pools. In H. Wainer (Ed.) Computerized Adaptive Testing: A Primer (2nd Ed., p.37-58). Mahwah, NJ:LEA.

Folk, V. & Smith, R. (2002) Models for Delivery of CBTs. In C. Mills, M. Potenza, J. Fremer, W. Ward (Eds.) *Computer-based testing: building the foundation for future assessments.* (p.41-66). Mahwah, NJ: LEA.

Follman, J. (1971). On the theory of achievement test items. Journal of Educational Measurement. 8(1), 57-59.

Foster, R. (July, 2010). Improve the output from a MCQ test item generator using statistical NLP. Paper presented at the 10th meeting of the IEEE

International conference on advanced learning technologies. Sousse, Tunisia.

Geerlings, H., van der Linden, W., & Glas, C. (2011). Modeling rule-based item generation. *Psychometrika, 76,* 337-359.

Gierl, M. & Lai, H. (2011) The role of item models in Automatic Item Generation. Paper presented at the Annual Meeting of the National Council on Measurement in Education. New Orleans, LA. April, 8-10.

Gierl, M., & Haladyna, T. (2012). *Automatic item generation: Theory and practice.* New York: Routledge.

Gierl, M., & Lai, H. (2012). Using weak and strong theory to create item models for automatic item generation: Some practical guidelines with examples. In M. J. Gierl & T. Haladyna (Eds.). *Automatic item generation: Theory and practice.* New York: Routledge.

Gierl, M., Alves, C., & Taylor-Majeau, R. (2010). Using the Attribute Hierarchy Method to make diagnostic inferences about examinees' skills in mathematics: An operational implementation of cognitive diagnostic assessment. International Journal of Testing, 10, 318-341.

Gierl, M., Alves, C., & Taylor-Majeau, R. (2010). Using the Attribute Hierarchy Method to make diagnostic inferences about examinees' skills in

mathematics: An operational implementation of cognitive diagnostic assessment. International Journal of Testing, 10, 318-341.

Gierl, M., Alves, C., Roberts, M., & Gotzmann, A. (April, 2009). Using judgments from content specialists to develop cognitive models for diagnostic assessments. Paper presented at the annual meeting of the National Council on Measurement in Education, San Diego, CA.

Gierl, M., Alves, C., Roberts, M., & Gotzmann, A. (April, 2009). Using judgments from content specialists to develop cognitive models for diagnostic assessments. Paper presented at the annual meeting of the National Council on Measurement in Education, San Diego, CA.

Gierl, M., Lai, H., & Breithaupt , K. (2012). Methods for creating and evaluating the item model structure used in automatic item generation. Paper presented at the annual meeting of the National Council on Measurement in Education, Vancouver, BC.

Gierl, M., Lai, H., & Turner, S. (2012). Using automatic item generation to create multiple-choice items for assessments in medical education. *Medical Education*, 46, 757-765.

Gierl, M., Zhou, J., & Alves, C. (2008). Developing a taxonomy of item model types to promote assessment engineering. *Journal of Technology,*

Learning, and Assessment, 7(2). Retrieved 1April, 2011, from http://www.jtla.org.

Glas, C. & van der Linden, W. (2006). Computerized Adaptive Testing with Item Cloning. *LSAC Computerized Testing Report*, 01-03.

Glas, C. (2006) Alternative Approaches to Updating Item Parameter Estimates in Tests with Item Cloning. *LSAC Research Report Series*. 03-01, 1-7.

Glas, C., & van der Linden, W. (2003). Computerized adaptive testing with item cloning. *Applied Psychological Measurement*, 27, 247-261.

Gorin, J. (2007). Test Construction and Diagnostic Testing. In J. P. Leighton & M. J. Gierl, Eds. Cognitive Diagnostic Assessment in Education: Theory and Practice. Cambridge University Press.

Gorin, J. S. (2007). Test Construction and Diagnostic Testing. In J. P. Leighton & M. J. Gierl, Eds. Cognitive Diagnostic Assessment in Education: Theory and Practice. Cambridge University Press.

Graesser, A., Rus, V., Cai, Z., & Hu, X. (2012). Question Answering and Generation. In P. McCarthy, & C. Boonthum-Denecke (Eds.), *Aplied Natural Language Processing: Identification, Investigation and Resolution* (p. 1-16). Hershey, PA: Information Science Reference.

Gütl, C, Lankmayr, K, Weinhofer, J and Höfler, M. "Enhanced Automatic Question Crea-tor – EAQC: Concept, Development and Evaluation of an

Automatic Test Item Creation Tool to Foster Modern e- Education" *The Electronic Journal of e-Learning Volume 9 Issue 1 2011, (pp23-38), available online at www.ejel.org*

Guttman, L. (1959). Introduction to facet design and analysis. *Acta Psychologica*, 15, 130–138.

Haladyna & Gierl (2012) In M. J. Gierl & T. Haladyna (Eds.). *Automatic item generation: Theory and practice.* New York: Routledge.

Haladyna, T. (1994). Developing and validation multiple-choice test items. Hillsdale, NJ: LEA.

Haladyna, T. (2012). Automatic Item Generation: A Historical Perspective. In M. J. Gierl & T. Haladyna (Eds.). *Automatic item generation: Theory and practice.* New York: Routledge.

Higgins, D. (2007). *Item Distiller: Text retrieval for computer-assisted test item creation. Educational Testing Service Research Memorandum* (RM-07-05). Princeton, NJ: Educational Testing Service.

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

Hoffman, R., & Lintern, G. (2006). Eliciting and representing the knowledge of experts. In K. Ericsson, N. Charness, P. Feltovich, & R. Hoffman. (Eds.). *Cambridge handbook of expertise and expert performance* (p. 203-222). New York: Cambridge University Press.

Holling, H., Bertling, J. , & Zeuch, N. (2009). Automatic item generation of probability word problems. *Studies in Educational Evaluation*, 35, 71–76.

Hornke, L. (2002). Item generation models for higher cognitive functions. In S. Irvine & P. Kyllonen (Eds.), Item generation for test development (p.159-178). Hillsdale, NJ: Erlbaum.

Huff, K. & Goodman, D. (2007). The Demand for Cognitive Diagnostic Assessment. In J. Leighton & M. Gierl (Eds.), Cognitive diagnostic assessment for education: Theory and applications. (p. 19–60). Cambridge, UK: Cambridge University Press.

Huff, K., Alves, C., Pellegrino, J. & Kaliski, P. (2012). Using evidence-centered design task models in automatic item generation. In M. J. Gierl & T. Haladyna (Eds.). *Automatic item generation: Theory and practice.* New York: Routledge.

Irvine, S., & Kyllonen, P. (2002). Item generation for test development. Hillsdale, NJ: Erlbaum.

Janssen, R. (2010). Modeling the effect of item designs within the Rasch model. In S. E. Embretson (Ed.), Measuring psychological constructs: Advances in model-based approaches (p. 227-245). Washington DC: American Psychological Association.

Janssen, R., Schepers, J., & Peres, D. (2004). Models with item and group predictors. In P. DeBoeck & M. Wilson (Eds.), *Explanatory item response models: A generalized linear and non-linear aproach* (p. 189-212). New York: Springer.

Janssen, R., Tuerlinkx, F., Meulders, M., & De Boeck, P. (2000). A hierarchical IRT model for criterion-referenced measurement. *Journal of Educational and Behavioral Statistics*, 25, 285–306.

Karamanis, N., Ha, L., & Mitkov, R. (2006). Generating multiple-choice test items from medical text: a pilot study. Proceedings of the Fourth International Natural Language Generation Conference (INLG '06). Association for Computational Linguistics. 111-113.

Kazi, H., Haddawy, P., Suebnukarn, S. (2012). Employing UMLS for Generating Hints in a Tutoring System for Medical Problem-Based Learning. *Journal of Biomedical Informatics.* 45(3). 557-565.

Klein Entink, R., Kuhn, J., Hornke, L., & Fox, J. (2009). Evaluating Cognitive Theory: A Joint Modeling Aproach Using Responses and Response Times. Psychological Methods,14(1), 54-75.

Kubinger, K. (2008). On the revival of the Rasch model-based LLTM: From constructing tests using item generating rules to measuring item administration effects. *Psychology Science Quarterly*, 50(3), 311-327.

LaDuca, A., Staples, W. I., Templeton, B., & Holzman, G. B. (1986). Item modeling procedures for constructing content-equivalent multiple-choice questions. *Medical Education*, 20, 53-56.

Lai, H., & Becker, K. (April, 2010). Using Artificial Neural Networks to Identify Enemy Item Pairs. Poster presented at the National Council for Measurement in Education. Denver, CO.

Lai, H., & Gierl, M. J. (April, 2012). Design Principles Required for Skills-Based Calibrated Item Generation. Paper presented at the annual meeting of the National Council on Measurement in Education. Vancouver, BC.

Lai, H., Alves, C., & Gierl, M. (June, 2009). Applying item model taxonomy for automatic item generation for CAT. Proceedings of the 2009 GMAC Conference on Computerized Adaptive Testing. Retrieved April 1, 2011 from www.psych.umn.edu/psylabs/CATCentral/

Lai, H., Alves, C., Zhou, J., & Gierl, M. (May, 2009). Using Automatic Item Generation to Address Current Test Development Issues. Paper presented at the annual meeting of the Canadian Society for Studies in Education, Ottawa, ON.

Lai, H., Gierl, M., & Alves, C. (April, 2010). Generating items under the assessment engineering framework. Paper presented at the annual meeting of the National Council on Measurement in Education. Denver, CO.

Lai, H., Turnbull, A. & Gierl, M. J. (April, 2009). Incorporating Auxiliary Information for Automatic Item Generation. Poster presented at 2009 American Education Research Association Annual Conference, Div. D Graduate Poster Session. San Diego, CA.

Leighton, J. & Gokiert, R. (April, 2005). The Cognitive Effects of Test Item Features: Informing Item Generation by Identifying Construct Irrelevant Variance. Paper presented at the annual meeting of the National Council on Measurement in Education. Montreal, Canada.

Leighton, J. P. & Gierl, M. J. (2007a). Defining and evaluating models of cognition used in educational measurement to make inferences about examinees' thinking processes. Educational Measurement: Issues and Practice, 26, 3-16.

Leighton, J. P., & Gierl, M. J. (2007b). Verbal reports as data for cognitive diagnostic assessment. In J. P. Leighton & M. J. Gierl (Eds.), Cognitive diagnostic assessment for education: Theory and aplications. (p. 146–172). Cambridge, UK: Cambridge University Press.

Leighton, J. P., & Gierl, M. J. (2011). *The learning sciences in educational assessment: The role of cognitive models.* Cambridge, UK: Cambridge University Press.

Lin, C., (June, 2004). ROUGE: A package for automatic evaluation of summaries. Proceedings of the ACL-04 Workshop. Philadelphia, PA.

Liu, B. (2009, June). SARAC: A Framework for Automatic Item Generation. Proceedings of the Ninth IEEE International Conference on Advanced Learning Technologies, p. 556-558.

Liu, X. (2010). *Using and Developing measurement instruments in science education: a Rasch modeling approach.* Charlotte, NC : IAP.

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

Luecht, R. (April, 2007). Assessment Engineering in Language Testing: From Data Models and Templates to Psychometrics. Paper presented at the annual

meeting of the National Council on Measurement in Education, Chicago, IL.

Luecht, R. (2012). An introduction to assessment engineering for automatic item generation. In M. J. Gierl & T. Haladyna (Eds.). *Automatic item generation: Theory and practice.* New York: Routledge.

Masters, J. (2010). *A comparison of traditional test blueprinting and item development to assessment engineering in a licensure context.* Doctoral Dissertation. University of North Carolina Greensboro. Greensboro, NC.

Millman, J., & Westman, R. (1989). Computer-assisted writing of achievement test items: toward a future technology. Journal of Educational Measurement, 26(2), 177-190.

Mislevy, R., Steinberg, L., & Almond, R. (2002). On the roles of task model variables in assessment design. In S. H. Irvine & P. C. Kyllonen (Eds.), Item generation for test development (p. 129-157). Mahwah, NJ: Lawrence Erlbaum.

Nake, F. (2009) Creativity in Algorithmic Art. Proceeding of the seventh ACM conference on Creativity and Cognition, 97-106.

National Research Council (2001). Knowing what the students know: the science and design of educational assessment. National Academy Press.

Nichols, P. (1994). A Framework for Developing Cognitively Diagnostic Assesments. Review of Educational Research, 64(4), 575-603.

Norman, G. (1988), Problem-solving skills, solving problems and problem-based learning. *Medical Education*, 22, 279–286.

Norman, G. and Eva, K. (2003). Doggie diagnosis, diagnostic success and diagnostic reasoning strategies: an alternative view. *Medical Education*, 37, 676–677.

Norman, G., Eva, K., Brooks, L., & Hamstra, S. (2006). Expertise in medicine and surgery. In *The Cambridge handbook of expertise and expert performance*. Cambridge, UK: Cambridge University Press. 393-353.

O'Brien, H. (2012, May 23). Eisenhower school records focus on role of ex-principal in testing irregularities. *Green Bay Press Gazette.* Retrieved from http://www.greenbaypressgazetted.com

O'Neil, L. (2012, June 13). Student brings 35-foot-long cheat sheet into exam, gets expelled. *CBC News.* Retrieved from http://www.cbc.ca

Parshall, C. (2002). Item development and pretesting in a CBT environment. In C. Mills, M. Potenza, J. Fremer & W. Ward (Eds.) Computer-based testing: Building the foundation for future assessments (p. 119-141). Mahwah, NJ: LEA.

Pellegino, J. (2002). Knowing what students know. *Issues in Science and Technology*, 19(2), 48-52.

R Development Core Team (2012). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Reiter, E. & Dale, R. (1997). Building Aplied Natural-Language Generation Systems. *Natural Language Engineering*, 3, 57-87.

Roid, G., & Haladyna, T. (1977). Measurement problems in the formative evaluation of instructional systems. *Improving Human Performance*, 6, 30-44.

Roid, G., & Haladyna, T.(1982*). A technology for test-item writing.* Maryland Heights, MO: Academic Press.

Rudner, L. (2010). Implementing the Graduate Management Admission Test Computerized Adaptive Test. In W. van der Linden & C. Glas (Eds.), *Elements of Adaptive Testing* (p. 151-165), New York, NY: Springer.

Rupp, A., Templin, J., & Henson, R. (2010). Diagnostic Measurement: Theory, Methods and Applications. New York, NY: Guilford.

Scalise, K., & Gifford, B. (2006). Computer-based assessment in e-learning: A framework for constructing “intermediate constraint” questions and

tasks for technology platforms. *Journal of Technology, Learning, and Assessment*, 4(6). Retrieved on May1, 2012 from http://www.jtla.org.

Schmeiser, C & Welch, C. (2006). Test Development in R. Brennan (Ed.) Educational measurement, 4th Edition. Westport, CT: Praeger.

Shu, Z., Burke, M. & Luecht, R. (April, 2010). Some quality control results of using a hierarchical Bayesian calibration system for assessment engineering task models, templates, and items. Paper presented at the Annual Meeting of the National Council on Measurement in Education, Denver, CO.

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

Sinharay, S., Johnson, M. S., & Williamson, D. M. (2003). Calibrating item families and summarizing the results using family expected response functions. *Journal of Educational and Behavioral Statistics, 28,* 295-313.

Snow, R. (1993). Construct validity and constructed-response tests. In R. Bennett & W. Ward (Eds.), Construction versus choice in cognitive measurement: Issues in constructed response, performance testing, and portfolio assessment (p. 45-60). Hillsdale, NJ: LEA.

Sternberg, R. & Nigro, G. (1980). Developmental Patterns in the Solution of Verbal Analogies. Child Development, 51, 27-38.

van den Bosch, A. , & Bouma, G. (2011). Interactive multi-modal question answering. Berlin, Springer.

van der Linden, W. (1998). Optimal assembly of psychological and educational tests. Applied Psychological Measurement, 22, 195-211.

Weiner, H., (2002). On the Automatic Generation of Test Items: Some Whens, Whys, and Hows, in S. H. Irvine & P. C. Kyllonen (Eds.), Item generation for test development . Mahwah, NJ: Erlbaum.

Weiss, D. (1982). Improving measurement quality and efficiency with adaptive test. Applied Psychological Measurement, 6, 473-492.

Welch, C. (2006). Item and Prompt Development in Performance Testing in S. Downing & T. Haladyna (Eds.) Handbook of Test Development. Mahwah, NJ: Lawrence Erlbaum.

Wendt, A., Kao, S., Gorham, J., & Woo, A. (2009). Developing item variants: An empirical study. In D. J. Weiss (Ed.), Proceedings of the 2009 GMAC Conference on Computerized Adaptive Testing. Retrieved April 1, 2011 from www.psych.umn.edu/psylabs/CATCentral/

Wild, F. (2011). lsa: Latent Semantic Analysis package for R. URL http://cran.r-project.org/web/packages/lsa/index.html

Zamost, S., Griffin, D., & Ansari, A. (2012, January 13). Exclusive: Doctors cheated on exams. *CNN Health.* Retrieved from http://www.cnn.com

Appendix A

1-Layer Hernia Model

```
<?xml version="1.0" encoding="UTF-8"?>
<Template>
  <Stem>
    <Value>A [[AGE]]-year-old [[Gender]] presented with a mass [[Pain]] in [[Location]]. It occured [[AcuityofOnset]]. On examination, the mass is [[PhysicalFindings]] and lab work came back with  [[WBC]]. Which of the following is the next best step?</Value>
  </Stem>
  <Variable name="WBC" type="String">
    <Value key="1">normal results</Value>
    <Value key="2">normal results</Value>
    <Value key="3">elevated white blood cell count</Value>
    <Value key="4">normal results</Value>
  </Variable>
  <Variable name="PhysicalFindings" type="String">
    <Value key="1">pertruding but with no pain</Value>
    <Value key="2">tender</Value>
    <Value key="3">tender and exhibiting redness</Value>
    <Value key="4">tender and reducible</Value>
  </Variable>
  <Variable name="AcuityofOnset" type="String">
    <Value key="1">a few months ago</Value>
    <Value key="2">a few hours ago</Value>
    <Value key="3">a few days ago</Value>
    <Value key="4">a few days ago after moving a piano</Value>
  </Variable>
  <Variable name="Location" type="String">
    <Value key="1">the left groin</Value>
    <Value key="2">right groin</Value>
    <Value key="3">the umbilicus</Value>
    <Value key="4">an area near a recent surgery</Value>
  </Variable>
  <Variable name="Pain" type="String">
    <Value key="1"/>
    <Value key="2">and intense pain</Value>
    <Value key="3">and severe pain</Value>
    <Value key="4">and mild pain</Value>
  </Variable>
  <Variable name="Gender" type="String">
    <Value key="1">man</Value>
    <Value key="2">woman</Value>
  </Variable>
  <Variable name="AGE" type="Number">
    <Range max="60.0" min="25.0" step="5.0"/>
    <Constraint key="1">([[WBC]]==[[PhysicalFindings]])</Constraint>
    <Constraint key="1">([[AcuityofOnset]]==[[Pain]])</Constraint>
    <Constraint key="1">([[WBC]]==[[Pain]])</Constraint>
  </Variable>
  <Option group="Key" type="String">
    <Value>ice applied to mass </Value>
  </Option>
  <Option group="Distractor" type="String">
    <Value>hernia repair</Value>
  </Option>
  <Option group="Distractor" type="String">
    <Value>reduction of mass</Value>
  </Option>
  <Option group="Distractor" type="String">
    <Value>exploratory surgery</Value>
  </Option>
</Template>
```

N-Layer Hernia Model

```
<?xml version="1.0" encoding="UTF-8"?>
<Template>
  <Stem>
    <Value>[[Sentence1]] [[Sentence2]] [[Sentence3]]</Value>
  </Stem>
  <Variable name="Sentence3" type="String">
    <Value key="1">What is the best next step?</Value>
    <Value key="2">Which one of the following is the best prognosis?</Value>
    <Value key="3">Given this information, what is the best course of action?</Value>
  </Variable>
  <Variable name="Sentence2" type="String">
    <Value key="1">On examination, the mass is [[PhysicalFindings]] and lab work came back with  [[WBC]].</Value>
    <Value key="2">Upon further examination, the patient had [[WBC]] and the mass is [[PhysicalFindings]].</Value>
    <Value key="3">With [[WBC]] and [[PhysicalFindings]] in the area, the patient is otherwise nominal.</Value>
    <Value key="4">There is [[PhysicalFindings]] in the [[Location]] and the patient had [[WBC]].</Value>
  </Variable>
  <Variable name="Sentence1" type="String">
    <Value key="1">A [[AGE]]-year-old [[Gender]] presented with a mass [[Pain]] in [[Location]]. It occured
[[AcuityofOnset]].</Value>
    <Value key="2">Patient presents with a mass [[Pain]] in [[Location]] from [[AcuityofOnset]]. The patient is a [[AGE]]-
year-old [[Gender]].</Value>
    <Value key="3">Patient complaints of a mass [[Pain]] in [[Location]] which has been a problem since
[[AcuityofOnset]].</Value>
    <Value key="4">A [[Gender]] was admitted with pain in the [[Location]] from [[AcuityofOnset]].</Value>
  </Variable>
  <Variable name="WBC" type="String">
    <Value key="1">normal vitals</Value>
    <Value key="2">normal vitals</Value>
    <Value key="3">elevated white blood cell count</Value>
    <Value key="4">normal vitals</Value>
  </Variable>
  <Variable name="PhysicalFindings" type="String">
    <Value key="1">pertruding but with no pain</Value>
    <Value key="2">tenderness</Value>
    <Value key="3">tender and exhibiting redness</Value>
    <Value key="4">tender and reducible</Value>
  </Variable>
  <Variable name="AcuityofOnset" type="String">
    <Value key="1">a few months ago</Value>
    <Value key="2">a few hours ago</Value>
    <Value key="3">a few days ago</Value>
    <Value key="4">a few days ago after moving a piano</Value>
  </Variable>
  <Variable name="Location" type="String">
    <Value key="1">the left groin</Value>
    <Value key="2">right groin</Value>
    <Value key="3">the umbilicus</Value>
    <Value key="4">an area near a recent surgery</Value>
  </Variable>
  <Variable name="Pain" type="String">
    <Value key="1"/>
    <Value key="2">and intense pain</Value>
    <Value key="3">and severe pain</Value>
    <Value key="4">and mild pain</Value>
  </Variable>
  <Variable name="Gender" type="String">
    <Value key="1">man</Value>
    <Value key="2">woman</Value>
  </Variable>
  <Variable name="AGE" type="Number">
    <Range max="60.0" min="25.0" step="5.0"/>
    <Constraint key="1">([[WBC]]==[[PhysicalFindings]])</Constraint>
    <Constraint key="1">([[AcuityofOnset]]==[[Pain]])</Constraint>
```

```
      <Constraint key="1">([[WBC]]==[[Pain]])</Constraint>
    </Variable>
    <Option group="Key" type="String">
      <Value>ice applied to mass </Value>
    </Option>
    <Option group="Distractor" type="String">
      <Value>hernia repair</Value>
    </Option>
    <Option group="Distractor" type="String">
      <Value>reduction of mass</Value>
    </Option>
    <Option group="Distractor" type="String">
      <Value>exploratory surgery</Value>
    </Option>
  </Template>
```

Appendix B

1-Layer Post-Operative Fever Model

```
<?xml version="1.0" encoding="UTF-8"?>
<Template>
  <Stem>
    <Value>A [[Age]]-year-old [[Gender]] has a [[Surgery]]. On post operative day [[Day]], the patient has a temperature
of [[Temperature]] C.[[Cue]] Which one of the following is the most likely diagnosis?</Value>
  </Stem>
  <Variable name="Cue" type="String">
    <Value key="1"/>
    <Value key="2"/>
    <Value key="3">Physical examination reveal a red and tender wound at the opening. </Value>
    <Value key="4"/>
    <Value key="5"/>
    <Value key="6">Physical examination reveal tenderness in the abdominal region with guarding and rebound.
</Value>
  </Variable>
  <Variable name="Surgery" type="String">
    <Value key="1">gastrectomy</Value>
    <Value key="2">right hemicholectomy</Value>
    <Value key="3">left hemicholectomy </Value>
    <Value key="4">appendectomy</Value>
    <Value key="5">laparoscopic cholecytectomy</Value>
  </Variable>
  <Variable name="Day" type="String">
    <Value key="1">3</Value>
    <Value key="2">2</Value>
    <Value key="3">4</Value>
    <Value key="4">3</Value>
    <Value key="5">4</Value>
    <Value key="6">6</Value>
  </Variable>
  <Variable name="Gender" type="String">
    <Value key="1">man</Value>
    <Value key="2">woman</Value>
  </Variable>
  <Variable name="Option3" type="String">
    <Value key="1">Urinary tract infection</Value>
    <Value key="2">Actelectasis</Value>
    <Value key="3">Wound infection</Value>
    <Value key="4">Pneumonia</Value>
    <Value key="5">Deep vein thrombosis</Value>
    <Value key="6">Deep space infection</Value>
  </Variable>
  <Variable name="Option2" type="String">
    <Value key="1">Urinary tract infection</Value>
    <Value key="2">Actelectasis</Value>
    <Value key="3">Wound infection</Value>
    <Value key="4">Pneumonia</Value>
    <Value key="5">Deep vein thrombosis</Value>
    <Value key="6">Deep space infection</Value>
  </Variable>
  <Variable name="Option1" type="String">
    <Value key="1">Urinary tract infection</Value>
    <Value key="2">Actelectasis</Value>
    <Value key="3">Wound infection</Value>
    <Value key="4">Pneumonia</Value>
    <Value key="5">Deep vein thrombosis</Value>
    <Value key="6">Deep space infection</Value>
  </Variable>
  <Variable name="Scenario" type="String">
    <Value key="1">Urinary tract infection</Value>
```

```
      <Value key="2">Atelectasis</Value>
      <Value key="3">Wound infection</Value>
      <Value key="4">Pneumonia</Value>
      <Value key="5">Deep vein thrombosis</Value>
      <Value key="6">Deep space infection</Value>
    </Variable>
    <Variable name="Temperature" type="Number">
      <Range max="38.5" min="38.0" step="0.5"/>
    </Variable>
    <Variable name="Age" type="Number">
      <Range max="70.0" min="40.0" step="30.0"/>
      <Constraint key="1">[[Scenario]]!=[[Option1]]</Constraint>
      <Constraint key="1">[[Scenario]]!=[[Option2]]</Constraint>
      <Constraint key="1">[[Scenario]]!=[[Option3]]</Constraint>
      <Constraint key="1">[[Option1]]!=[[Option2]]</Constraint>
      <Constraint key="1">[[Option2]]!=[[Option3]]</Constraint>
      <Constraint key="1">[[Option1]]!=[[Option3]]</Constraint>
      <Constraint key="1">([[Cue]]==[[Scenario]])</Constraint>
      <Constraint
key="1">(([[Scenario]]!=6)||(([[Scenario]]==6)&amp;amp;&amp;amp;&amp;amp;(([[Surgery]]==3)||([[Surgery]]==4))))</Constraint>
      <Constraint
key="1">(([[Scenario]]!=1)||(([[Scenario]]==1)&amp;amp;&amp;amp;&amp;amp;([[Surgery]]&amp;lt;4)))</Constraint>
      <Constraint key="1">(([[Scenario]]!=2)||(([[Scenario]]==2)&amp;amp;&amp;amp;&amp;amp;([[Surgery]]!=4)))</Constraint>
      <Constraint
key="1">(([[Scenario]]!=3)||(([[Scenario]]==3)&amp;amp;&amp;amp;&amp;amp;([[Surgery]]&amp;gt;3)))</Constraint>
      <Constraint
key="1">(([[Scenario]]!=4)||(([[Scenario]]==4)&amp;amp;&amp;amp;&amp;amp;([[Surgery]]&amp;lt;4)))</Constraint>
      <Constraint
key="1">(([[Scenario]]!=5)||(([[Scenario]]==5)&amp;amp;&amp;amp;&amp;amp;(([[Surgery]]==1)||([[Surgery]]==5))))</Constraint>
      <Constraint key="1">([[Day]]==[[Scenario]])</Constraint>
    </Variable>
    <Option group="Key" type="String">
      <Value>[[Scenario]]</Value>
    </Option>
    <Option group="Distractor" type="String">
      <Value>[[Option1]]</Value>
    </Option>
    <Option group="Distractor" type="String">
      <Value>[[Option2]]</Value>
    </Option>
    <Option group="Distractor" type="String">
      <Value>[[Option3]]</Value>
    </Option>
  </Template>
```

N-Layer Post-Operative Fever Model

```
<?xml version="1.0" encoding="UTF-8"?>
<Template>
  <Stem>
    <Value>[[Situation]] [[Scenario.PhysicalExamination]] [[Question.cue]]</Value>
  </Stem>
  <Variable name="Question.cue" type="String">
    <Value key="1">Which one of the following is the most likely diagnosis for this patient?</Value>
    <Value key="2">Which one of the following is the best next step for this patient?</Value>
  </Variable>
  <Variable name="Situation" type="String">
    <Value key="1">A [[Age]]-year-old woman has a [[Surgery]].  On post operative day [[Scenario.TimingOfFever]] she
has a temperature of 38.5 C.</Value>
    <Value key="2">A [[Age]]-year-old patient who had a [[Surgery]] had a fever of 38.6 C on post-operative day
[[Scenario.TimingOfFever]].</Value>
  </Variable>
  <Variable name="Scenario.PhysicalExamination" type="String">
    <Value key="1"/>
    <Value key="2"/>
    <Value key="3">Physical examination reveal a red and tender wound at the opening. </Value>
    <Value key="4"/>
    <Value key="5"/>
    <Value key="6">Physical examination reveal tenderness in the abdominal region with guarding and rebound.
</Value>
  </Variable>
  <Variable name="Surgery" type="String">
    <Value key="1">gastrectomy</Value>
    <Value key="2">right hemicholectomy</Value>
    <Value key="3">left hemicholectomy </Value>
    <Value key="4">appendectomy</Value>
    <Value key="5">laparoscopic cholecytectomy</Value>
  </Variable>
  <Variable name="Scenario.TimingOfFever" type="String">
    <Value key="1">3</Value>
    <Value key="2">2</Value>
    <Value key="3">4</Value>
    <Value key="4">3</Value>
    <Value key="5">4</Value>
    <Value key="6">6</Value>
  </Variable>
  <Variable name="Option3.man" type="String">
    <Value key="1">Antibiotics</Value>
    <Value key="2">Mobilize </Value>
    <Value key="3">Reopen wound</Value>
    <Value key="4">Antibiotics</Value>
    <Value key="5">Anti coagulation</Value>
    <Value key="6">Drainage</Value>
  </Variable>
  <Variable name="Option3.diag" type="String">
    <Value key="1">Urinary tract infection</Value>
    <Value key="2">Actelectasis</Value>
    <Value key="3">Wound infection</Value>
    <Value key="4">Pneumonia</Value>
    <Value key="5">Deep vein thrombosis</Value>
    <Value key="6">Deep space infection</Value>
  </Variable>
  <Variable name="Option2.man" type="String">
    <Value key="1">Antibiotics</Value>
    <Value key="2">Mobilize </Value>
    <Value key="3">Reopen wound</Value>
    <Value key="4">Antibiotics</Value>
    <Value key="5">Anti coagulation</Value>
    <Value key="6">Drainage</Value>
  </Variable>
```

```
  <Variable name="Option2.diag" type="String">
    <Value key="1">Urinary tract infection</Value>
    <Value key="2">Actelectasis</Value>
    <Value key="3">Wound infection</Value>
    <Value key="4">Pneumonia</Value>
    <Value key="5">Deep vein thrombosis</Value>
    <Value key="6">Deep space infection</Value>
  </Variable>
  <Variable name="Option1.man" type="String">
    <Value key="1">Antibiotics</Value>
    <Value key="2">Mobilize </Value>
    <Value key="3">Reopen wound</Value>
    <Value key="4">Antibiotics</Value>
    <Value key="5">Anti coagulation</Value>
    <Value key="6">Drainage</Value>
  </Variable>
  <Variable name="Option1.diag" type="String">
    <Value key="1">Urinary tract infection</Value>
    <Value key="2">Actelectasis</Value>
    <Value key="3">Wound infection</Value>
    <Value key="4">Pneumonia</Value>
    <Value key="5">Deep vein thrombosis</Value>
    <Value key="6">Deep space infection</Value>
  </Variable>
  <Variable name="Scenario.man" type="String">
    <Value key="1">Antibiotics</Value>
    <Value key="2">Mobilize </Value>
    <Value key="3">Reopen wound</Value>
    <Value key="4">Antibiotics</Value>
    <Value key="5">Anti coagulation</Value>
    <Value key="6">Drainage</Value>
  </Variable>
  <Variable name="Question.Scenario" type="String">
    <Value key="1">[[Scenario.diag]]</Value>
    <Value key="2">[[Scenario.man]] </Value>
  </Variable>
  <Variable name="Question.Op1" type="String">
    <Value key="1">[[Option1.diag]]</Value>
    <Value key="2">[[Option1.man]] </Value>
  </Variable>
  <Variable name="Question.Op2" type="String">
    <Value key="1">[[Option2.diag]]</Value>
    <Value key="2">[[Option2.man]] </Value>
  </Variable>
  <Variable name="Question.Op3" type="String">
    <Value key="1">[[Option3.diag]]</Value>
    <Value key="2">[[Option3.man]] </Value>
  </Variable>
  <Variable name="Scenario.diag" type="String">
    <Value key="1">Urinary tract infection</Value>
    <Value key="2">Atelectasis</Value>
    <Value key="3">Wound infection</Value>
    <Value key="4">Pneumonia</Value>
    <Value key="5">Deep vein thrombosis</Value>
    <Value key="6">Deep space infection</Value>
  </Variable>
  <Variable name="Age" type="Number">
    <Range max="58.0" min="43.0" step="4.0"/>
    <Constraint key="1">[[Scenario]]!=[[Option1]]</Constraint>
    <Constraint key="1">[[Scenario]]!=[[Option2]]</Constraint>
    <Constraint key="1">[[Scenario]]!=[[Option3]]</Constraint>
    <Constraint key="1">[[Option1]]!=[[Option2]]</Constraint>
    <Constraint key="1">[[Option2]]!=[[Option3]]</Constraint>
    <Constraint key="1">[[Option1]]!=[[Option3]]</Constraint>
    <Constraint
key="1">((([[Scenario]]!=6)||((([[Scenario]]==6)&amp;amp;&amp;amp;((([[Surgery]]==3)||([[Surgery]]==4))))</Constraint>
```

```
      <Constraint
key="1">(([[Scenario]]!=1)||(([[Scenario]]==1)&amp;amp;&amp;amp;([[Surgery]]&amp;lt;4)))</Constraint>
      <Constraint key="1">(([[Scenario]]!=2)||(([[Scenario]]==2)&amp;amp;&amp;amp;([[Surgery]]!=4)))</Constraint>
      <Constraint
key="1">(([[Scenario]]!=3)||(([[Scenario]]==3)&amp;amp;&amp;amp;([[Surgery]]&amp;gt;3)))</Constraint>
      <Constraint
key="1">(([[Scenario]]!=4)||(([[Scenario]]==4)&amp;amp;&amp;amp;([[Surgery]]&amp;lt;4)))</Constraint>
      <Constraint
key="1">(([[Scenario]]!=5)||(([[Scenario]]==5)&amp;amp;&amp;amp;(([[Surgery]]==1)||([[Surgery]]==5))))</Constraint>
    </Variable>
    <Option group="Key" type="String">
      <Value>[[Question.Scenario]]</Value>
    </Option>
    <Option group="Distractor" type="String">
      <Value>[[Question.Op1]]</Value>
    </Option>
    <Option group="Distractor" type="String">
      <Value>[[Question.Op2]]</Value>
    </Option>
    <Option group="Distractor" type="String">
      <Value>[[Question.Op3]]</Value>
    </Option>
</Template>
```