

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/286441120>

Automatic Generation of Valid and Equivalent Assessment Instruments

Conference Paper · April 2014

DOI: 10.1007/978-3-662-44188-6_13

CITATIONS

0

READS

131

1 author:



[Shilpi Banerjee](#)

International Institute of Information Technology Bangalore

8 PUBLICATIONS 14 CITATIONS

SEE PROFILE

Automatic Generation of Valid and Equivalent Assessment Instruments

Shilpi Banerjee and Chandrashekar Ramanathan

International Institute of Information Technology, Bangalore
shilpi.banerjee@iiitb.ac.in,rc@iiitb.org

Abstract. Assessment is an integral part of the student's learning process. It is essential to design good quality assessment instruments. Traditional assessment instruments rely on blueprints and random selection of items from the item bank. In this paper, we propose a method of generating valid assessment patterns automatically that satisfies the constraints provided by the instructor. Moreover, we generate a large item bank by using automatic item generation, which helps in designing equivalent, multiple assessment instruments to keep a check in plagiarism. We present an assessment instrument for a course on Design of Digital Systems that is developed using the above approach. The effectiveness of the assessment instrument is ensured by considering the validity of items.

1 Introduction

The essential factors for designing a good quality assessment instrument are - **Validity** - A valid assessment is one which measures what it is intended to measure[1]. To ensure the validity aspect, it is essential to design assessments in alignment with the stated aims for a course.

Assessment specification - It is a detailed description for the instrument, often called a blueprint, that specifies the number of proportion of items that assess each content, skill, etc[2].

Alternate(Equivalent) forms - Two or more versions of a assessment that are considered interchangeable(equivalent), when they measure the same constructs in the same ways and are intended for the same purpose[2]. For large scale assessments, having equivalent forms of instrument helps in preventing plagiarism.

The conventional way of designing an assessment instrument, that satisfies the above discussed factors demands devotion of enormous time and energy for the instructor as the process is highly labour-intensive and error-prone. With growing usage of technology for teaching and learning process, there is a need to explore the development of automated methods in the area of assessment. This paper discusses a unique approach to generate a good quality assessment instrument automatically. Section 2 presents the related work and the concepts related to the proposed approach used in this paper, Section 3 presents the proposed approach, Section 4 shows the result for an example and Section 5 summarizes with concluding remarks.

2 Research Background and Related Work

The existing systems for automatic generation of instrument can be categorised broadly into two categories based on pattern generation and item generation, respectively. The following sub-sections contain additional information on these two categories. Some of the terminology used in these sections is defined in Table 1.

2.1 Automatic Assessment Pattern Generation

Assessment blueprints[3] guide the development of instrument by ensuring the right distribution of number of items for all the attributes by creating a pattern for the assessment instrument. The effort related to the creation of blueprints does not remain manually feasible for large number of assessments for a course. Several methods[4][6] have been designed to address this issue for two attributes. If the number of attributes are more than 2, the generation of assessment instrument starts becoming unmanageably complex if done manually. Out of various approaches used to automate the generation of instrument when the number of attributes are more than 2, random extraction and retrospective testing are mostly used these days by the existing software tools[5]. Each of these approaches lacks an important component of assessment, that is alignment of assessment items with the competency and the validity of assessment items is not taken into consideration while creating the assessment pattern.

2.2 Automatic Assessment Item Generation

The systems designed in [5][6] uses a database of items for creating instrument and focus was on the composition of only one instrument. [8] uses randomization technique for organising multiple sets of assessment instrument, but it does not guarantee their equivalence. Because of limited size of item bank, creating alternate forms of instrument is challenging. Large scale assessments demands multiple sets of equivalent instrument. The size of item bank can be increased by using automatic item generation.

3 Proposed Approach

In this paper, we have discussed a way of designing an assessment instrument using three dimensional matrix from the assessment pattern generator and the database of items generated by using automatic item generation. The assessment instrument generation comprises of three essential blocks as discussed in subsequent sections.

3.1 Alignment Pattern Generation

This module takes care of **validity** aspect of the instrument. The correct estimate of student's learning can be estimated by including assessment items in the assessment instrument which are aligned with the competency[1]. Each and every competency focuses over one or many cognitive levels. The assessment item for any competency should be taken from the same or lower cognitive level to which it is aligned. The items are termed to be valid only if it satisfies the alignment constraint(ac). Alignment constraint is provided by the instructor as a input to this module. We get a two dimensional matrix($matrix_{ac}$) at the output which shows the relation between the competency and the cognitive level it belongs to. The instrument will use this matrix for selecting valid items.

3.2 Assessment Pattern Generation

Assessment pattern generation provides the functionality to fulfil a given **assessment specification** and gives a three dimensional matrix that shows the marks allotted to items belonging to each of the attribute categories. The maximum possible number of categories is equal to product of number of competencies(m), number of cognitive levels(n) and number of assessment item types(p). Invalid combinations of competency and cognitive level are filtered out by the alignment pattern generator. Total marks(M_T) for the instrument and weight constraints($w_c = \{W_C, W_{CL}, W_{AT}\}$) for competency, cognitive level and assessment item type are given by the instructor as a input to this module. If W_C, W_{CL}, W_{AT} are given by $W_C = \{W_{C_1} \cdots W_{C_m}\}, W_{CL} = \{W_{CL_1} \cdots W_{CL_n}\}, W_{AT} = \{W_{AT_1} \cdots W_{AT_p}\}$, then

$$\sum W_C = \sum W_{CL} = \sum W_{AT} = M_T \quad (1)$$

The real challenge is to select appropriate number of assessment items from each of these sub categories of assessment item attributes such that equation(1) get satisfied always. **Algorithm 1** presents the pseudo code for assessment pattern generation. As a first round of explanation for the pseudo code of the algorithm, we mention the following points: **satisfy** is designed as a *depth-first state-space* search and **satisfy** uses a *branch and bound* approach to limit search along infeasible paths in the *search tree*. The size of the state space is a function of m , n and p . State space denotes the set of all the candidate assessment patterns. If there are 5 competencies, 6 cognitive levels and 3 assessment item types, the maximum number of possible categories are $5 * 6 * 3 = 90$ and the state space will have an upper bound of 3^{90} , if number of allowed values for every cell of assessment pattern is 3. As the number of these attributes increases, the state space starts becoming unmanageably large. The algorithm discussed in [6][7] explores all possible states in any case. In our approach, we have used branch and bound algorithm which searches the state space in such a way that there are early evidences available, if exploring in any particular direction is not effective in producing result. The search space for assessment pattern is reduced with the help of intermediate and final constraint. **Intermediate constraint** is checked

for all cell positions except the last one. The sum of values placed in the cells populated so far should be equal to or less than the weight constraints. **Final constraint** is checked for the last cell position. The sum of values placed in the cells populated so far (all the cells in the matrix, since this is the last cell) should be equal to the weight constraints. For a partially filled matrix, the sum s of values in all cells populated so far should be less than or equal to the weight constraint wc . If $s < wc$, then the remaining cells can be populated with appropriate values so as to make the final $s = wc$. If $s = wc$, then the remaining cells may be filled with zeros. However, if $s > wc$, then there cannot be any values with which the remaining cells could be filled so as to make the final $s = wc$. Hence, for all cells except the last one, we use intermediate constraint. For the last cell of the matrix, s must be equal to wc as there are no cells left to make $s = wc$, in case $s < wc$.

3.3 Automatic Item Bank Generation

Automatic item generation (AIG) was developed to address the increasing demand for assessment items for creating **alternate forms** of the instrument. In the item bank, item templates instead of items are stored along with respective tags for competency, cognitive level and assessment item types. The item templates are written in a manner indicating the integration of cognitive level associated with the competency. Item templates are constructed by using two essential elements, stem and variables. By varying the variables of an item in a systematic manner, AIG can be used to create multiple items in an iterative way as discussed in **Algorithm 2**. All the items generated from the same template are equivalent in nature as the value set for the variable have similar characteristics. Two different values are considered to be equivalent if they address the same content domain. The similarity of values for generating equivalent items can be decided by the instructor. We can design different difficulty item templates in a single cognitive level by selecting variable with different characteristics. The algorithm for generating assessment instrument is discussed in **Algorithm 3**.

4 Result - Example Generation

We have developed item templates for the course, Design of Digital Systems. A formative assessment instrument is generated for first two competencies and the sample is included in the paper. The competencies considered are as mentioned in Table 2. The assessment items are validated by using the alignment pattern generator. Table 3 shows valid set of assessment items ($\{AI\}$) and their tags. The assessment item types which were considered are selection type and supply type. All selection type items are of 1 mark while supply type items were designed for 1 and 2 marks. Total marks (M_T) for the instrument is 20 while the weight constraint for 2 competencies, 3 cognitive levels and 2 assessment item types are $W_C = \{W_{C1}, W_{C2}\} = \{8, 12\}$, $W_{CL} = \{W_{CL1}, W_{CL2}, W_{CL3}\} = \{8, 8, 4\}$, $W_{AT} = \{W_{AT1}, W_{AT2}\} = \{6, 14\}$. The maximum possible number of categories

for this combination is $2 * 3 * 2 = 12$. Assessment pattern finds the number of items chosen from each of these categories. Table 4 shows an assessment pattern for the weight constraints mentioned above. Table 5 shows an instance for automatic generation where a template is used to design 8 equivalent items. The tag $C_1CL_2AT_2$ denotes that the item template belongs to the first competency, second cognitive level(understand) and second assessment item type(supply type item). Table 6 and 7 gives assessment instruments for the above mentioned assessment specification.

5 Conclusion

The proposed assessment instrument provides generalization related to alignment question[1] which discusses about the importance of alignment of competency with assessment. Alignment pattern generator ensures the validity of items for assessment which helps in providing the evidence of how well students have learned what the instructor intends them to learn. Assessment specification pattern is automatically developed using depth first branch and bound search algorithm. This algorithm helps in finding the first solution that meets the weight constraint. The complexity of this algorithm increases exponentially as a function of the number of attributes. Equivalent sets of assessment instrument can be designed using the optimized assessment pattern. Our future work will focus on evaluating alternate form reliability and looking for evidences of valid assessment, when the instrument is offered to students.

Table 1. Definition Of Terminology Used

Term	Definition
Assessment instrument	Assessment instrument is used for summative or formative assessment. It is designed for specified maximum marks and it addresses a chosen set of competencies.
Assessment item(AI)	A general term referring to a single statement, question, exercise, problem, or task on a assessment or evaluative instrument for which the assessment taker is to select or construct a response, or to perform a task. [2].
Competency(C)	It is a detailed description of what students will be able to do when they complete a unit of instruction[4].
Cognitive level(CL)	Bloom's taxonomy divides the cognitive domain into 6 levels which are remember(CL_1) understand(CL_2), apply(CL_3), analyse(CL_4), evaluate(CL_5) and create(CL_6) [4]
Assessment item type(AT)	The assessment items can be selection type or supply type.
Attribute	It is used to characterize an assessment item for various competency, cognitive level and assessment item type.

Algorithm 1 satisfy - For Creating Assessment Pattern

```

satisfy(pos, matrixac, matrixwc) {pos- cell position, matrixac- 2D matrix(size
= len(SC)*len(SCL)), matrixwc- 3D matrix(size = len(SC)*len(SCL))
*len(SAT))}
if matrixac[pos] = -1 then
    Evaluate all possible values for matrixwc [pos]{valid cell position}
    if pos ≠ lastcell then
        if intermediateConstraint(matrixwc[pos]) = True then
            satisfy(nextcellpos, matrixwc)
        end if
    else if pos = lastcell then
        if finalConstraint(matrixwc[pos]) = True then
            return True {solution that satisfies wc can be found in matrixwc}
        end if
    end if
else if matrixac[pos] = 0 then
    matrixwc[pos] = 0 {invalid cell position}
end if
return matrixwc

```

Algorithm 2 generateItems - For Creating Item Bank From Item Templates

```

generateItems(tag-template file, variable-value file)
for all templates in {T} do
    {{T} ← Set of templates for a item template bank}
    Extract {VarT} {{VarT} ← Set of variables for a template}
    for all variables in {VarT} do
        {{Valvar} ← Set of values for a variable}
        Extract {Valvar}
        Find cross product of values present for each of the variable
        Create all items for template by replacing variables by the values found by cross
        product
        Write created items with associated tags in item-bank file
    end for
end for
return item-bank

```

Algorithm 3 createInstrument - For Creating Assessment Instrument

```

createInstrument (matrixwc, item-bank, matrixai) {matrixai - 3D matrix(size =
len(SC)*len(SCL)) *len(SAT))}
list ← empty list for storing items per index
for all index of matrixwc do
    Tn ← value in the index {index- tag for items} {value- marks allotted to that
    index}
    list[index] ← Randomly choose items from item-bank having the same index such
    that marks allotted to all of them sum to Tn
    matrixai[index] ← list[index]
end for
return matrixai

```

References

1. Lorin W. Anderson, David R. Krathwohl: A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Pearson,(2000)
2. Joint Committee on Standards for Educational and Psychological Testing of the AERA,APA, and NCME, Standards for educational and psychological testing. Washington DC: American Educational Research Association, (1999)
3. Chan Kan Kan "Using Test Blueprint in Classroom Assessments: Why and How", International Association for Educational Assessment, (2010)
4. Dimple V.Paul, Shankar B.Naik, Priyanka Rane, Jyoti D.Pawar, "Use of an Evolutionary Approach for Question Paper Template Generation",IEEE, Fourth International Conference on Technology for Education, (2012).
5. Song Yan, Yang Guoxing, "A Genetic Algorithm of Test Paper Generation", IEEE, Eighth International Conference On Computer Science and Education, (2013).
6. Vaibhav M. Kale, Arvind W. Kiwelekar, "An Algorithm for Question Paper Template Generation in Question Paper Generation System", IEEE, International Conference on Technological Advances in Electrical, Electronics and Computer Engineering, (2013)
7. Vijay Krishan Purohit, Abhijeet Kumar, Asma Jabeen, Saurabh Srivastava, R H Goudar, Shivanagowda, Sreenivasa Rao, "Design of Adaptive Question Bank Development and Management System", IEEE, International Conference on Parallel, Distributed and Grid Computing, (2012)
8. Nor Shahida bt Mohd Jamail, Abu Bakar Md Sultan, "Shuffling Algorithms for Automatic Generator Question Paper System", Computer and Information Science, (2010)

Table 2. Competencies

SNo	Competencies	Cognitive level
C_1	Understand the nature of logic expressions written in terms of logical functions(AND, OR, NOT, NAND, NOR, X-OR, X-NOR)	Understand
C_2	Simplify logical expressions using Boolean theorems.	Apply

Table 3. Valid Set Of Assessment Items

	Remember	Understand	Apply	Analyse	Evaluate	Create
C_1	$\{AI\}_{C_1CL_1}$	$\{AI\}_{C_1CL_2}$	-	-	-	-
C_2	$\{AI\}_{C_2CL_1}$	$\{AI\}_{C_2CL_2}$	$\{AI\}_{C_2CL_3}$	-	-	-

Table 4. Automatic Assessment Pattern Generation- Output From Algorithm **satisfy**

		Assessment Pattern		
C_1	AT_1	$\{AI\}_{C_1CL_1AT_1} = 2$	$\{AI\}_{C_1CL_2AT_1} = 1$	$\{AI\}_{C_1CL_3AT_1} = 0$
	AT_2	$\{AI\}_{C_1CL_1AT_2} = 3$	$\{AI\}_{C_1CL_2AT_2} = 2$	$\{AI\}_{C_1CL_3AT_2} = 0$
C_2	AT_1	$\{AI\}_{C_2CL_1AT_1} = 1$	$\{AI\}_{C_2CL_2AT_1} = 1$	$\{AI\}_{C_2CL_3AT_1} = 1$
	AT_2	$\{AI\}_{C_2CL_1AT_2} = 2$	$\{AI\}_{C_2CL_2AT_2} = 4$	$\{AI\}_{C_2CL_3AT_2} = 3$

Table 5. Automatic item generation using item templates- Output from algorithm **generateItems**

Tag	AI template	Variable	Value set	Assessment Items
$C_1CL_2AT_2$	A logic circuit has three input and one output variables. The output variable is at $C_1CL_2AT_{2,a}$ when two or more inputs are at $C_1CL_2AT_{2,b}$. Write the truth table and realize using only $C_1CL_2AT_{2,c}$ gates.	$C_1CL_2AT_{2,a}$	{logic 1, logic 0}	A logic circuit has three input and one output variables. The output variable is at logic 1 when two or more inputs are at logic 1. Write the truth table and realize using only NAND gates.
		$C_1CL_2AT_{2,b}$	{logic 1, logic 0}	A logic circuit has three input and one output variables. The output variable is at logic 1 when two or more inputs are at logic 1. Write the truth table and realize using only NOR gates.
		$C_1CL_2AT_{2,c}$	{NAND, NOR}	A logic circuit has three input and one output variables. The output variable is at logic 1 when two or more inputs are at logic 0. Write the truth table and realize using only NAND gates.
				A logic circuit has three input and one output variables. The output variable is at logic 1 when two or more inputs are at logic 0. Write the truth table and realize using only NOR gates.
				A logic circuit has three input and one output variables. The output variable is at logic 0 when two or more inputs are at logic 1. Write the truth table and realize using only NAND gates.
				A logic circuit has three input and one output variables. The output variable is at logic 0 when two or more inputs are at logic 0. Write the truth table and realize using only NOR gates.
				A logic circuit has three input and one output variables. The output variable is at logic 0 when two or more inputs are at logic 0. Write the truth table and realize using only NOR gates.

Table 6. Automatically generated assessment instrument- Set 1- Output from algorithm **createInstrument**



Number of marks allotted per category of {AI}	Tag	Item	Marks
{AI} $_{C_1CL_1AT_1} = 2$	$C_1CL_1AT_1$	A circuit containing only AND and NOT gates must be a Combinational circuit. (A) True (B) False	1
	$C_1CL_1AT_1$	The Boolean expression for a 3-input AND gate is (A) $X = (ABC)'$ (B) $X = ABC$ (C) $X = A + B + C$ (D) $X = (A + B + C)'$	1
{AI} $_{C_1CL_1AT_2} = 3$	$C_1CL_1AT_2$	What is positive logic?	1
{AI} $_{C_1CL_2AT_1} = 1$	$C_1CL_1AT_2$	Draw the logic symbol, write the boolean expression and construct the truth table for the X-OR gate.	2
	$C_1CL_2AT_1$	Choose the gate that represents the function of the logic gate system below. (a) NAND gate (b) X-OR gate (c) NOR gate (d) X-NOR gate 	1
{AI} $_{C_1CL_2AT_2} = 2$	$C_1CL_2AT_2$	A logic circuit has three input and one output variables. The output variable is at logic '1' when two or more inputs are at logic '1'. Write the truth table and realize using only NAND gates.	2
{AI} $_{C_2CL_1AT_1} = 1$	$C_2CL_1AT_1$	Identify the correct statement for Demorgan's theorem (A) $(X+Y)' = X'Y'$ (B) $(X+Y)' = X'+Y'$ (C) $(XY)' = X'.Y'$ (D) $(XY)' = X'.Y'$	1
{AI} $_{C_2CL_1AT_2} = 2$	$C_2CL_1AT_2$	State Commutative law.	1
	$C_2CL_1AT_2$	Write the truth table for the following boolean expression $Y=A+B'C$	1
{AI} $_{C_2CL_2AT_1} = 1$	$C_2CL_2AT_1$	When simplified with Boolean Algebra $x(x+y)(x+z)$ simplifies to (A) x (B) $x + x(y+z)$ (C) $x(1+yz)$ (D) $x + yz$	1
{AI} $_{C_2CL_2AT_2} = 4$	$C_2CL_2AT_2$	Implement boolean expression for Ex-NOR using NAND gates.	2
	$C_2CL_2AT_2$	Complement the expression $(AB)' + A' + AB$	2
{AI} $_{C_2CL_3AT_1} = 1$	$C_2CL_3AT_1$	Evaluating $x = A'B + C(AD)'$ using the convention A = False and B = False gives (A) CD' (B) 0 (C) 1 (D) $C'D$	1
{AI} $_{C_2CL_3AT_2} = 3$	$C_2CL_3AT_2$	Prove the following Boolean identity using the laws of Boolean Algebra and implement the simplified equation using basic gates. $(x + xy') + xy + x(y' + z)$	3

Table 7. Automatically generated assessment instrument- Set 2- Output from algorithm **createInstrument**

Number of marks allotted per category of {AI}	Tag	Item	Marks
{AI} $_{C_1CL_1AT_1} = 2$	$C_1CL_1AT_1$	A circuit containing only OR and NAND gates must be a Sequential circuit. (A) True (B) False	1
	$C_1CL_1AT_1$	The Boolean expression for a 3-input NAND gate is (A) $X = (ABC)'$ (B) $X = ABC$ (C) $X = A + B + C$ (D) $X = (A + B + C)'$	1
{AI} $_{C_1CL_1AT_2} = 3$	$C_1CL_1AT_2$	What is negative logic?	1
{AI} $_{C_1CL_2AT_1} = 1$	$C_1CL_1AT_2$	Draw the logic symbol, write the boolean expression and construct the truth table for the NAND gate.	2
	$C_1CL_2AT_1$	Choose the gate that represents the function of the logic gate system below.(a) NAND gate (b) X-OR gate (c) NOR gate (d) X-NOR gate 	1
{AI} $_{C_1CL_2AT_2} = 2$	$C_1CL_2AT_2$	A logic circuit has three input and one output variables. The output variable is at logic '1' when two or more inputs are at logic '0'. Write the truth table and realize using only NOR gates.	2
{AI} $_{C_2CL_1AT_1} = 1$	$C_2CL_1AT_1$	Identify the correct statement for Demorgan's theorem (A) $(X+Y)' = X'Y'$ (B) $(X+Y)' = X'+Y'$ (C) $(XY)' = X'.Y'$ (D) $(XY)' = X'.Y'$	1
{AI} $_{C_2CL_1AT_2} = 2$	$C_2CL_1AT_2$	State Associative law.	1
	$C_2CL_1AT_2$	Write the truth table for the following boolean expression $Y=AB'+A'C$	1
{AI} $_{C_2CL_2AT_1} = 1$	$C_2CL_2AT_1$	When simplified with Boolean Algebra $x(x+y)(x+z)$ simplifies to (A) x (B) $x + x(y+z)$ (C) $x(1+yz)$ (D) $x + yz$	1
{AI} $_{C_2CL_2AT_2} = 4$	$C_2CL_2AT_2$	Implement boolean expression for Ex-OR using NOR gates.	2
	$C_2CL_2AT_2$	Complement the expression $(A' + B + C')(A' + B + C)$	2
{AI} $_{C_2CL_3AT_1} = 1$	$C_2CL_3AT_1$	Evaluating $x = A'B + C(AD)'$ using the convention A = True and B = False gives (A) CD' (B) 0 (C) 1 (D) $C'D$	1
{AI} $_{C_2CL_3AT_2} = 3$	$C_2CL_3AT_2$	Prove the following Boolean identity using the laws of Boolean Algebra and implement the simplified equation using basic gates. $(x' + y'z)(yz' + x')(y' + z')[x' + (y'+z)']$	3