Revision of the AIG Software Toolkit: A Contribute to More User Friendliness and Algorithmic Efficiency

Sebastian Kucharski[1], Gregor Damnik[2], Florian Stahr[1] and Iris Braun[1]
[1]*Computer Networks Group, Faculty for Computer Science, Technische Universität Dresden, Germany*
[2]*Center for Teacher Education and Educational Research, Technische Universität Dresden, Germany*
{*sebastian.kucharski, gregor.damnik, florian.stahr, iris.braun*}*@tu-dresden.de*

Keywords: Automatic Item Generation, AIG, Assessment, Cognitive Model, Item Model.

Abstract: The traditional way of constructing items to assess learning is time-consuming because it requires experts to perform the labor-intensive task of creating legions of items by hand. The Automatic Item Generation (AIG) approach aims to streamline this process by having experts not formulate individual items, but rather create highly structured models that can be used by software to automatically generate them. This requires two types of software components. First, an editor that allows experts to specify these models. Second, a generator that processes the specified models and generates the intended items. The elaboration of these components must address the following challenges. The former must be usable for the definition of complex knowledge models while the corresponding modeling process should be easy to understand. The latter should be able to process these models in a reasonable time. Thus, the goal is to overcome both challenges by defining and conceptualizing the use of a model representation that is easy to understand and efficient to process. Therefore, we present a new AIG software toolkit in relation to our previous work, which addresses these challenges by introducing a new representation approach - the layered-model-approach. The toolkit shall be evaluated in terms of usability and efficiency.

1 INTRODUCTION

The creation of items for the assessment of learning is a very time-consuming and resource-intensive process ((Gierl et al., 2012; Braun et al., 2022; Baum et al., 2021)). The reason for this is that the development of (test) items, requires experts who have to deal with several steps. They have to define the learning objectives in a subject area, write the items to meet those objectives, administer the test, score the test, and announce the results. While research has shown that test administration and scoring can easily be supported by the help of testing software (computer-based testing; e.g., (Gierl and Haladyna, 2012)), especially item writing is still often done by hand for each individual item. As a result, the individually written items are often problematic in terms of their standardization and comparability. This is not only due to the unlinked creation process, but also to the lack of analysis and revision after an assessment or learning situation has taken place. For example, items are rarely statistically analyzed for their ability to differentiate between learners and revised accordingly to improve them before they are added to a larger item pool. In this paper, we will introduce Automatic Item Generation (AIG; (Gierl et al., 2012; Braun et al., 2022; Baum et al., 2021; Damnik et al., 2018; Embretson, 2013; Embretson and Yang, 2006)), a process that is significantly different from the traditional way creating items. In addition, we will show how our new AIG item editor works and the progress we have made from our old software ((Braun et al., 2022; Baum et al., 2021)). Finally, we will discuss how items can be drawn from a larger item pool by using an algorithmic approach.

2 THE PROCESS OF AUTOMATIC ITEM GENERATION

In AIG, experts do not write individual items as in traditional item creation; rather, they define a representation of the subject area that is highly structured and standardized. This representation is called a *cognitive model* (Leighton and Gierl, 2011) in the AIG process. To create the cognitive model, experts must

Kucharski, S., Damnik, G., Stahr, F. and Braun, I.
Revision of the AIG Software Toolkit: A Contribute to More User Friendliness and Algorithmic Efficiency.
DOI: 10.5220/0011982000003470
In *Proceedings of the 15th International Conference on Computer Supported Education (CSEDU 2023) - Volume 2*, pages 410-417
ISBN: 978-989-758-641-5; ISSN: 2184-5026

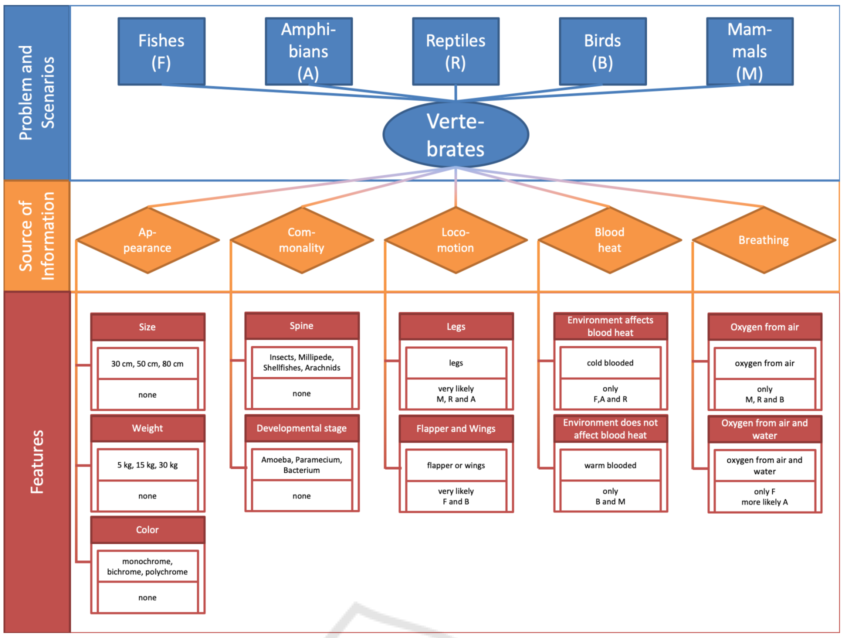


Figure 1: A cognitive model about different kinds of vertebrates.

define typical problems they face in the subject area and the information they need to solve those problems. Figure 1 shows a cognitive model of the subject area biology or more precisely of the topic *kinds of vertebrates*.

To generate items with the AIG process, experts first describe the problem of *kinds of vertebrates* by naming all five forms in the cognitive model: *fishes*, *amphibians*, *reptiles*, *birds*, and *mammals*. Then, they add three sources of information that are needed at a minimal level to decide which kind of vertebrate is given in a particular example. They name these sources: *locomotion*, *blood heat*, and *breathing*. This type of sources of information is called *case-specific*. Furthermore, they add two sources of information with their features that are often given in examples but are not really necessary to decide which kind of vertebrates is given: *appearance* and *commonality*. This kind of sources of information is called *generic*. Finally, the experts define the elements that each source of information or, more precisely, each feature can have and how these features are related to the different kinds of vertebrates. For example, they mention that mammals, reptiles, and birds breathe air through their lungs while fish and amphibians can (also) breathe from water through their gills.

Once the cognitive model is complete, the experts create an *item model* that can be used as a template for all items by defining the item stem, the sources of information and the sources of information of the features, and the options. The options are necessary because this model was created for single and multiple-choice items. Next, some sources of information are blanked out and the AIG software creates a set of items by combining the cognitive model and the item model. These items are then stored in an item bank, which supports the scoring of the test and the reporting of results. Figure 2 shows the item model mentioned above. Blanks are represented by [[...]] in the item stem. The sources of information are in capital letters and the elements are listed behind them. Figure 2 also shows two examples of the items generated for the problem *kinds of vertebrates*.

3 REQUIREMENT FOR A NEW AIG SOFTWARE TOOLKIT

We presented a software toolkit that contains the mentioned software for the definition of the described models and the use of these models for the generation of items in (Braun et al., 2022). This toolkit was developed in the context of the *AMCS-AIG* project and was intended for the generation of items for the Audience Response System (ARS) *AMCS* (Auditorium Mobile Classroom Service) (Braun et al., 2018). It included two components - a *model generator* and an *item generator*.

The model generator component, which we call the *AMCS-AIG editor*, is a web-based editor that can be used to define cognitive models and item models. Cognitive models in the AMCS-AIG editor consist of

Stem	If there is an animal with [[LOCOMOTION]], which is [[BLOOD HEAT]] and breaths [[BREATHING]]. The animal is normally approximately [[SIZE]] long. Which kind of animal is it?
Elements	LOCOMOTION: legs, flapper or wings BLOOD HEAT: cold blooded, warm blooded BREATHING: oxygen from air, oxygen from air and water SIZE: 30 cm, 50 cm, 80 cm
Options	Fish, Amphibian, Reptile, Bird, Mammal

If there is an animal with legs, which is warm blooded and breaths oxygen from air. The animal is normally approximately 30 cm long. Which kind of animal is it?

Amphibian
Reptile
Bird
Mammal *
Fish

If there is an animal with flapper or wings, which is warm blooded and breaths oxygen from air. The animal is normally approximately 30 cm long. Which kind of animal is it?

Amphibian
Reptile
Bird *
Mammal
Fish

Figure 2: The item model and item examples belonging to the cognitive model of different kinds of vertebrates.

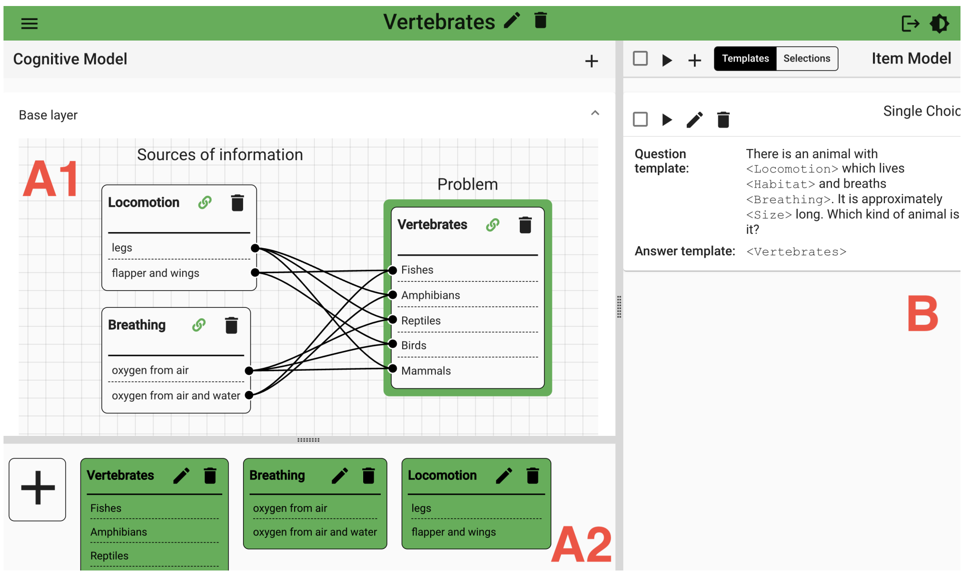
classes, *nodes*, and *edges* and can be defined either graphically or textually using a well-defined XML- or JSON-based description format. Classes, which correspond to sources of information, can be used to group nodes, which then represent features. Edges define connections between these elements and can also be grouped to restrict the valid paths through the cognitive model graph. These restrictions are implemented during the traversal of the graph by applying the following rule - *once a grouped edge is passed, only edges corresponding to previously passed edge groups can be passed.* Item models are built from an interrogative clause (i.e., the item stem), where the classes from the cognitive model are referenced using angle brackets.

The second component (i.e., the item generator), which we call *AIG Item Generator*, is responsible for the actual item generation. To do this, first, all possible nodes (i.e., features) are inserted into the corresponding reference gaps of the item model to generate all possible items, regardless of their validity or invalidity with respect to the conditions which are defined by the cognitive model. Then, the invalid items are removed by checking if the graph can be traversed using only the nodes referenced by the considered item without violating the described traversal rule.

Although these two components are convenient for creating and editing of small models and generating the corresponding items, a user evaluation revealed two major drawbacks for the processing of more complex models. First, we measured a processing time of up to several minutes for generating items for these models. This prevents the applicability for real-time scenarios. Thus, although items can be generated for complex models, this process cannot be interactive in the sense of allowing the user to review the generated items and dynamically modify parts of these underlying models. Second, as soon as the definition of the cognitive model requires the specification of complex branches in the corresponding graph, the use of the editor becomes difficult due to the appearance of the user interface and the representation of all information in a single graph, which thus quickly becomes confusing.

For these reasons, a revision of the components of the AIG toolkit was planned. The main objectives of the revision and the corresponding research and conception process were the following three.

1. Adapt the model editor to the current state of user interface-related research.
2. Conceptualize and implement a model representation format which is easy to understand even when it is used for the development of complex cognitive models.
3. Conceptualize and implement a more efficient algorithm for the generation of items for complex models.



Question template: There is an animal with `<Locomotion>` which lives `<Habitat>` and breaths `<Breathing>`. It is approximately `<Size>` long. Which kind of animal is it?

Answer template: `<Vertebrates>`

Figure 3: Representation of the vertebrates related cognitive model and item model in AME.

4 A NEW AIG SOFTWARE TOOLKIT

With regard to the revision goals specified in section 3, we have created two software components that correspond to the established ones presented in (Braun et al., 2022). The revised model generator that replaces the former *AMCS-AIG editor* is called *AIG Model Editor* (AME) and is described in section 4.1. The item generator that replaces the former *AIG Item Generator* is called *Item Generator using SAT* (IGuS) and is introduced in section 4.2.

4.1 Model Editor

The revised model editor, which we call *AIG Model Editor* (AME), is a web application with a user interface that is divided into two areas, as shown in figure 3. The area *B* can be used to develop item models. An item model consists of one or more item templates, which can be of different types. The type of item template defines what kind of items are generated during the processing of the considered template. There are three different types - *single choice*, *multiple choice* and *matching*. The sources of information, whose features are varied during the generation process, are referenced using angle brackets.

The area *A* (i.e., *A1* and *A2*) can be used to develop cognitive models. The sources of information and their corresponding features are defined in the area *A2* and can be added to the cognitive model graph in the area *A1* using drag & drop. The dependencies between the defined features are defined graphically by creating connections between the ports associated with each feature within the source of information node, as shown in figure 3.

4.1.1 The Layered Model Approach

The main idea of AME's approach to cognitive model representation, which we call the *layered model approach*, is to use separate graphs to represent connections upon multiple sources of information, rather than using a single graph with grouped edges to describe these connections throughout that graph. Therefore, the former complex graph (Braun et al., 2022) is replaced by multiple less complex graphs that are encapsulated in so-called *layers*. Each layer contains exactly one graph and has a type that determines how this graph is constructed. There are two types of layers.

1. *Base layers*, of which each cognitive model contains exactly one, are described in section 4.1.2.
2. Second, condition layers, of which each cognitive model can contain multiple or none, are described in section 4.1.3.

A conceptual example of a cognitive model with one base layer and one condition layer is shown in figure 4.

4.1.2 The Base Layer

Within the graph of the base layer of a cognitive model (e.g., area *A* in figure 4), the problem of this

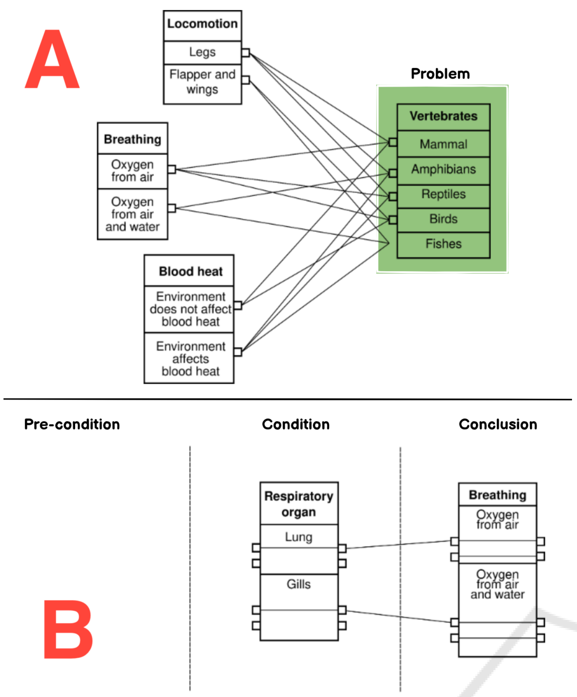


Figure 4: Representation of an extended vertebrates-related model using the layered model approach.

model is specified and the direct connections between the corresponding scenarios and the features of case-specific sources of information that directly infer a restriction of possible scenarios are defined. If there are no dependencies between sources of information and thus connections upon multiple sources of information and scenarios, the base layer and the definition of the sources of information described above represent the entire cognitive model. As soon as there are dependencies between sources of information, meaning the selection of one feature of a source of information restricts the possible selections of features of another source of information, additional condition layers are required to represent these dependencies.

4.1.3 The Condition Layers

Condition layers describe the dependencies upon multiple sources of information by specifying the dependencies for the features of one source of information in a separate layer. Therefore, the source of information whose dependencies are to be further described, is defined as the *condition* of the layer. In addition, the layer contains one or more *conclusion* and several or no *pre-condition* sources of information. The features of the pre-conditions can be connected to the ingoing ports of the condition. The features of the conclusions can be connected to the outgoing ports of the condition. For a fully specified condition layer, each outgoing port of the condition defines a constraint, which can be considered as a logical implication with a premise and a conclusion. The premise is constructed by creating a conjunction (i.e., a logical AND relation) from the corresponding feature and all features that have an edge to the ingoing port that is related to the considered outgoing port. The conclusion is constructed by creating a disjunction (i.e., a logical OR relation) from all features of the conclusion source of information that have an edge to the considered outgoing port. If there is more than one conclusion source of information, the mentioned disjunction combines the conjunctions of the features that are part of the paths through the conclusion sources of information.

In summary, AME is a modern web application that can be used to define cognitive models and item models for the AIG process. For the definition of the cognitive model, the newly conceptualized layered model approach is used. The main idea of this approach is to use one graph to specify the relationships between features and scenarios encapsulated in a so-called *base layer* and additional graphs to define dependencies upon multiple sources of information in so-called *condition layers*. The advantage of this approach is that cognitive models with extensive branching scenarios can be modeled without having to encapsulate all information in a single graph, which quickly becomes confusing. As a result, even complex cognitive models can be clearly represented and easily modeled.

4.2 Item Generator

The revised item generator, which we called *Item Generator using SAT* (IGuS), is a web service that can be invoked using a REST API. IGuS is based on the main idea of representing a cognitive model by a logical formula which we called the *cognitive model expression* and using a SAT solver for the determination of valid compilations of features of case-specific sources of information and scenarios. How this cognitive model expression is derived is explained in section 4.2.1. How this expression is used to automatically generate items is described in section 4.2.2.

4.2.1 Cognitive Model Expression

To derive a cognitive model expression from a given cognitive model, first, a boolean variable is assigned to each feature of a case-specific source of information and to each scenario. Then, for each condition defined by the cognitive model, a boolean implication term is constructed. If a variable is set to *TRUE* in such a term, it means that the represented feature or scenario can occur in a generated item together with all other features or scenarios whose variables are set to *TRUE* in the considered term, but not together with the features or scenarios whose variables are set to *FALSE*. For example, the condition that mammals, amphibians and reptiles use legs for locomotion and fishes and birds use flapper and wings, is represented by the following boolean term:

$$(Legs \Rightarrow (Mammals \lor Reptiles \lor Amphibians)) \land \\ (Flapper\ and\ Wings \Rightarrow (Fishes \lor Birds)) \tag{1}$$

This term represents the conditions that if a vertebrate uses either flapper and wings or legs for locomotion, it must be of the corresponding type, but nevertheless it cannot be used for the item generation yet. The reason for this is that if only this term would be considered, the variable *Legs* and the variable *Flapper and Wings* could be set to *TRUE* at the same time, suggesting that the compilation of legs, flapper and wings, mammals, reptiles, amphibians, fishes and birds is valid. Thus, another boolean term is needed to enforce the implicit condition of the cognitive model that only one feature of a source of information and one scenario of the defined problem can be used in an item. For the locomotion example the following terms are needed.

$$(\neg Legs \land Flapper\ and\ Wings) \lor \\ (Legs \land \neg Flapper\ and\ Wings) \tag{2}$$

$$\begin{aligned} &(Fishes \land \neg Amphibians \land \neg Reptiles \\ &\quad \land \neg Birds \land \neg Mammals) \lor \\ &(\neg Fishes \land Amphibians \land \neg Reptiles \\ &\quad \land \neg Birds \land \neg Mammals) \lor \\ &(\neg Fishes \land \neg Amphibians \land Reptiles \\ &\quad \land \neg Birds \land \neg Mammals) \lor \\ &(\neg Fishes \land \neg Amphibians \land \neg Reptiles \\ &\quad \land Birds \land \neg Mammals) \lor \\ &(\neg Fishes \land \neg Amphibians \land \neg Reptiles \\ &\quad \land \neg Birds \land Mammals) \end{aligned} \tag{3}$$

In summary, the conjunction of boolean implication terms for each condition that is defined by the cognitive model and boolean terms to enforce that for the problem and each source of information only one scenario or respectively one feature per item can be selected, leads to a logical representation of the whole cognitive model, which we call the *cognitive model expression*. Each variable assignment, for which this expression can be evaluated to *TRUE*, represents a valid compilation of case-specific features and scenarios of the represented cognitive model.

4.2.2 Item Generation

The item generation is done in two steps. First, the computation of the valid feature compilations using the defined cognitive model expression. Second, the actual creation of the items by inserting the features and scenarios of the determined compilations into the corresponding gaps of the item templates of the item model. The SAT solver library *Sat4j* (Le Berre and Parrain, 2010) is used to determine all valid feature compilations. To be processed by this library, the cognitive model expression is first transformed into the *CNF* format and then into the *DIMACS-CNF* format.

Once the valid compilations of scenarios and features of case-specific sources of information has been determined, the actual item creation process is performed. For each item template, the first step is to determine in which gaps scenarios or features of case-specific sources of information and in which gaps features of generic sources of information must be inserted. Then, the number of compilations is extended by combining each possible compilation of features of generic sources of information with each valid compilation of scenarios and features of case-specific sources of information. Finally, the required items are created by inserting the scenarios and features of each resulting compilation into the corresponding item template gaps.

In summary, IGuS is a web service that can be used to automatically generate items defined by a given item model, taking into account the conditions defined by a corresponding cognitive model. The generation is performed in the following two steps. First, the cognitive model is transformed into one boolean expression and a SAT solver is used to determine all valid compilations of features of case-specific sources of information. Second, the features and scenarios of these compilations are inserted into the item templates of the item model to generate the required items. This approach has the advantage that the computational complexity of these steps is inversely proportional to the frequency with which they need to be performed. The first step, is computationally expensive, but the computation result can be reused as long as the cognitive model does not change. Thus, it only needs to be done once to generate a large number of items. The second step must be performed for each combination of item template and determined compilation (i.e., for each item), but is computationally inexpensive. In addition, a highly optimized SAT solver is used to determine valid compilations, so that even the processing of complex models takes a few seconds at most.

5 ITEM SELECTION

Once all possible items have been generated for a given item model and the corresponding cognitive model, a mechanism is needed to select the generated items in a comprehensible way. The reason for this is that rarely are all items needed at once, but often only certain subsets of these items are required, while other item subsets could be used to dynamically replace them. Which items are selected for a particular subset depends on the use case for which the generated items are intended. We have defined the following two use cases for the utilization of automatically generated items:

1. *Exam* - Items are generated for a task of an exam. For this use case the items have to differ as much as possible with regard to their appearance to effectively prevent cribbing, but have to be as equal as possible with regard to the tested knowledge to ensure the equality of opportunity for all examinees.
2. *Individual* - Items are generated for individual studying or training for example in the range of exam preparation. For this use case the items have to differ as much as possible with regard to the tested knowledge, to ensure that the learner works with the complete learning matter, and can also differ with regard to their appearance to detract the recognition factor.

5.1 Implementation

The differentiation between items derived from a specific item model can be described by the used features of case-specific sources of information, features of generic sources of information and scenarios that were inserted into the gaps of the mentioned item model during the generation process as described in section 2. To make these differences measurable, we have defined two quantities - d_{cs} and d_g. d_{cs} is the distance between two features with respect to the used features of case-specific sources of information and scenarios, and d_g is the distance between two features with respect to the used features of generic sources of information. As an example, consider the following item template for the cognitive model shown in figure 1.

There is an animal with [[LOCOMOTION]] which breaths [[BREATHING]]. The animal is normally approximately [[SIZE]] long and has a weight of [[WEIGHT]]. Which kind of animal is it? [[VERTEBRATE]]

For this item template the following two items could be generated.

I1. There is an animal with *legs* which breaths *oxygen from air*. The animal is normally approximately *80 cm* long and has a weight of *30 kg*. Which kind of animal is it? *Mammal*

I2. There is an animal with *flapper or wings* which breaths *oxygen from air*. The animal is normally approximately *50 cm* long and has a weight of *30 kg*. Which kind of animal is it? *Bird*

For these two items, d_{cs} is 2 because they differ with respect to the scenario and the source of information *LOCOMOTION* but not with respect to the source of information *BREATHING*. d_g is 1 because they differ with regard to the source of information *SIZE* but not with regard to the source of information *WEIGHT*.

These two values can be utilized to select items for the defined use cases. For the *INDIVIDUAL*-selection use case the value of d_{cs} between the selected items is tried to be maximized. So first a random start item is chosen. Second, the item with the highest d_{cs} value with respect to the first item is selected. If there are multiple items with the same d_{cs} value, the item out of these with the highest d_g value with respect to the first item is selected. If there are multiple items with the same d_g value, the second item is selected at random. For the third value, the procedure is the same with regard to the quantity prioritization, but now both already selected items are taken into account. When considering multiple items, the calculated distance values are weighted to prevent a selection where multiple items are similar and only one item has a large distance value to all other items. Thus, when the remaining items are ranked for selection, the number of items to which a considered item has the largest distance is counted and the selection is made on that basis, rather than summing the distance values to all other items and selecting the item with the highest value. Subsequent items are selected in the same way until the required number of items is reached.

For the *EXAM*-selection use case the value of d_{cs} between the selected items is tried to be minimized and the value of d_g between the selected items is tried to be maximized. Therefore, first a random starting element is chosen. Second, the item with the lowest d_{cs} value with respect to the first item is selected. If there are multiple items with the same d_{cs} value, the item out of these with the highest d_g value with regard to the first item is selected. If there are multiple items with the same d_g value, the second item is selected at random. The selection of the following items works analogous with regard to the quantity prioritization and uses the same weighting mechanism for determining distances to multiple items as described for the *INDIVIDUAL*-selection use case.

6 CONCLUSION

Although the development of the *AMCS-AIG editor* and the *AIG Item Generator* in the context of the *AMCS-AIG* project (Braun et al., 2022) greatly simplified the generation of large item sets in less time than the traditional creation of such sets, the evaluation of these components revealed several drawbacks. We described these drawbacks in detail, defined the goals we pursued during the revision of our AIG software toolkit and presented the conceptual results of our investigation. We then presented our revised AIG software toolkit, the implementation of which took into account the aforementioned conceptual findings. This toolkit consists of two components. First, the *AIG Model Editor* - a graphical editor that can be used to define AIG models and that uses a newly introduced representation approach called the *layered model approach*. This approach is based on the idea of representing cognitive models in multiple graphs instead of one graph to improve the usability of the editor and to flatten the learning curve for the AIG model creation process. Second, the *Item Generator using SAT* - an item generator that represents cognitive models by boolean equations and which uses a SAT solver for the actual generation process. The main advantage of this component is that, due to the short processing time, the user can request the modeled items in almost real time. We showed how items can be generated for a simple cognitive model, with the intention of describing how this process works for more complex models during the presentation. Finally, we described why the selection of generated items is necessary to support different learning scenarios, specified such scenarios, and presented a mechanism how this selection can be done with respect to the considered scenarios. Future work will include completing the evaluation of our new software toolkit in terms of usability and efficiency.

REFERENCES

Baum, H., Damnik, G., Gierl, M. J., and Braun, I. (2021). A shift in automatic item generation towards more complex tasks. In *INTED2021 Proceedings*, pages 3235–3241. IATED.

Braun, I., Damnik, G., and Baum, H. (2022). Rethinking assessment with automatic item generation. In *INTED2022 Proceedings*, pages 1176–1180. IATED.

Braun, I., Kapp, F., Hara, T., Kubica, T., and Schill, A. (2018). Amcs (auditorium mobile classroom service)–an ars with learning questions, push notifications, and extensive means of evaluation. In *CEUR Workshop Proceedings*, volume 2092.

Damnik, G., Gierl, M., Proske, A., Körndle, H., and Narciss, S. (2018). Automatische erzeugung von aufgaben als mittel zur erhöhung von interaktivität und adaptivität in digitalen lernressourcen. In *E-Learning Symposium 2018*, pages 5–16. Universitätsverlag Potsdam.

Embretson, S. and Yang, X. (2006). 23 automatic item generation and cognitive psychology. *Handbook of statistics*, 26:747–768.

Embretson, S. E. (2013). Generating abstract reasoning items with cognitive theory. In *Item generation for test development*, pages 251–282. Routledge.

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

Gierl, M. J., Lai, H., and Turner, S. R. (2012). Using automatic item generation to create multiple-choice test items. *Medical education*, 46(8):757–765.

Le Berre, D. and Parrain, A. (2010). The sat4j library 2.2, system description. *Journal on Satisfiability, Boolean Modeling and Computation To appear*.

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