


Automatic Generation of Problem-Solving Assessment Items Through Domain Model Variations

Inga M. Saatz ^a

Fachbereich Informatik, University of Applied Sciences and Arts Dortmund, Emil-Figge Strasse 42, Dortmund, Germany

Keywords: Item Generation, Problem-Solving, Domain Model, Assessment.

Abstract: In computer science education, variations in the application contexts in modelling and programming tasks enhance the development of problem-solving skills. This results in a demand for a vast training and testing corpus of open questions with varying domain models usable in online and offline assessments. This paper proposes a two-step workflow for the automatic generation of items with varied domain models. First, experiences are provided about using the generated test corpora in formative assessments in computer science courses in higher education, especially for problem-based questions.

1 INTRODUCTION

In an online assessment, an interaction between learners during the examinations could not be sufficiently excluded, mainly if web-based test tools by a learning management system are used or programming tasks should be done within the examination. One solution is the provision of individual assessment tasks for each student of similar difficulty levels. This individualised approach requires the development and quality assurance of a vast pool of test items before the examination.

In computer science education, however, students learn problem-solving skills, such as analysing and modelling real-life scenarios, implementing the models in computer programs, and testing the implemented programs. Subsequently, students show their ability to understand and apply their knowledge in the context of practical problems by using different domains (Ferreira et al., 2018).


This leads to the research question of how to develop items for measuring problem-solving skills in computer science education using models for multiple domains, which would effectively prevent students from cheating in online examinations. However, support for varying domain models in automatic item generation is missing, especially for items addressing problem-solving skills.

This paper proposes a two-step workflow for the automatic generation of items with varied domain models applicable to higher education in computer science.

This paper is structured as follows: The second paragraph outlines the related work, whereas the following paragraph presents the proposed two-step workflow. The fourth paragraph discusses the first experiences using the proposed item generation workflow, followed by a discussion section. The paper concludes with a summary and an outlook.

2 RELATED WORK

Closed question types are most frequently used in automatic item generation, such as single- and multiple-choice, free-text, and fill-in-the-blank questions. In contrast, open-ended questions are less used in educational assessments (Circi, Hicks, Sikali, 2023). All these question types consist of a reading passage with a context description followed by the question to be answered. Furthermore, closed question types contain items, distractors, and the correct answers (Gierl et al. 2021). In mathematical and engineering education, various numerical values might be sufficient to create different items. In computer science, however, the automatic generation of variations in the task's domain model description

^a <https://orcid.org/0000-0002-7371-806X>

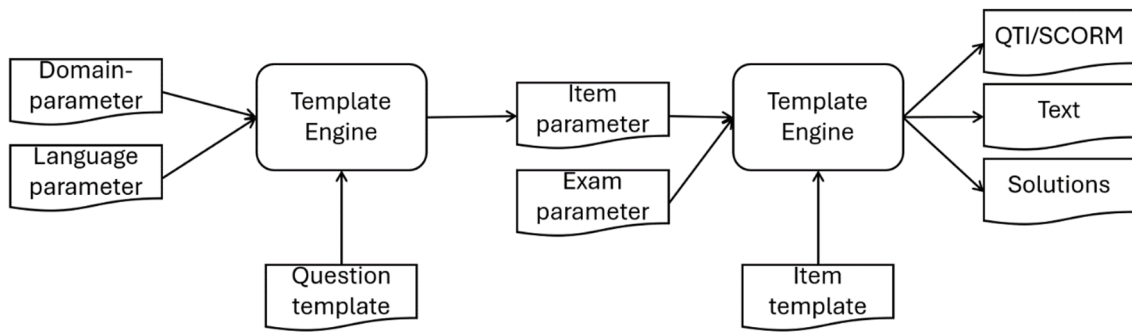


Figure 1: Automatic item generation workflow.

is necessary for programming and modelling tasks, especially to address problem-solving skills. Foulonneau and Ras (Foulonneau and Ras, 2013) generated test questions from Semantic resources in the Semantic Web for educational purposes. This approach could generate tasks for testing factual and contextual knowledge using the content data described by the corresponding Resource Description Framework (RDF). This leads to question types with content variations, for example, in questions for naming the capitals of different countries. However, typical computer science modelling and programming tasks focus on describing structures and metadata of the content data and not the content itself. The corresponding domain model contains entity and condition descriptions, such as ‘A [country] must have one [capital]’. Therefore, modelling and programming tasks are more related to the higher skill levels of application and analysis in Bloom’s revised taxonomy (Anderson, Krathwohl, 2001).

Furthermore, modifications of the domain model affect the task description text. Xiao et al. (Xiao et al., 2018), for example, reported on automatically generated multiple-choice tasks for language learning by selecting appropriate prepositions in the stem and distractors generation. They created a feature list and syntax tree for the example sentences to identify the blank’s position and to classify the task’s difficulty. However, a domain model modification affects the whole task description text. For example, when replacing the words in brackets in the condition example above with the words ‘woman’ and ‘child’, the obligation (‘must’) does not fit. It should be changed to an optional condition to fit better into real-world scenarios. Furthermore, the change of verbs and nouns in a sentence may imply changes to the word forms, too, especially in languages relying on grammatical cases, such as German. Thus, a particular language model is needed to provide correct grammar and sentence structure.

In a literature review, Setianwan, Hidayah, and Kusumawardami found three kinds of evaluation methods (expert-based, student-based, and mathematical model-based evaluation) for automatic item datasets (Setianwan, Hidayah, and Kusumawardami, 2022). In expert-based evaluation, the item datasets are evaluated, for example, regarding objectivity, answerability, and correctness.

3 TASK GENERATION FOR PROGRAMMING EDUCATION

3.1 Workflow for the Automatic Item Generation

The proposed approach separates content, domain, and layout information to generate multiple question types based on various domain models. This resulted in a two-step item generation workflow depicted in Figure 1.

In the first step of the workflow, question templates are rendered with domain parameters by a template engine. Question templates comprise all content data needed for creating a question-type item. This includes, for example, the reading passages, context description, question text, selection items, solution texts, and links to figures to be embedded in questions or solutions. Furthermore, language declination parameters for the domain models could be provided. In this example, the language parameters could be used to adapt the indefinite articles according to the starting character of the domain parameter. A rule-based extension of the templating engine is used to declinate articles, nouns, and adjectives in German text passages. Through this templating approach, long and complex sentences could be used in the item texts. The output of the first step of the generation workflow is a JSON file with item parameters, one for each domain model.

In the workflow’s second step, the item template is rendered with the item and examination parameters. The examination parameters comprise the assessment-specific data, such as date, course, duration, and participant data (if necessary). The created output depends on the used item template and the purpose of the items. For an online assessment, for example, the QTI (Question and Test Interoperability specification) item template creates an import file for the learning management system. Text item templates are used to generate LaTeX files for a paper-based assessment. Furthermore, separate solution resources could be generated to support the faculty staff in evaluating the student solutions to the individualised assessment tasks. Generated solution hints could consist of lists of keywords. These lists are used to check student problem-based solutions automatically.

3.2 Example 1: Modelling Item

One area of problem-based assessment tasks in computer science education is modelling tasks according to described scenarios. As an example, the question text of a task regarding formulating a database query looks like as follows:

Consider the entities [Word1] and [Word2] and the condition 'A [Word1] [Obligation] have [Cardinality] [Word2]'. Implement corresponding relational model in SQL with appropriate attributes and constraints.

In this case, the domain parameter file contains a set of domain parameters for each domain model, as shown in Table 1.

Table 1: Examples of domain parameters.

word1	word2	obligation	cardinality
country	capital	must	one
mother	child	could	many

The first item in Table 1 calls for creating one relational table with a constraint, which checks the presence of a capital name. The second item in Table 1 is modelled by creating two relational tables, one of which references the other. The resulting (simplified) keyword lists are depicted in Table 2.

Table 2: Example of solution hints.

Word2	Modelling	Keyword lists
capital	one table	CREATE TABLE country capital NOT NULL
child	two tables	CREATE TABLE mother CREATE TABLE child mother REFERENCES

Through this two-step rendering process, structurally identical items are generated. The difficulty level of each item is similar, depending on the domain parameters used. Structural variations between the items could be achieved by varying the corresponding question templates.

3.3 Example 2: Programming Item

In computer science education, programming tasks are used to test problem-solving skills. Fill-in-the-blank question types could be used to test knowledge about the programming syntax while providing parts of the problem solution, as shown in the following item template example.

Given are the relational tables
[mother] (id, name) and
[child] (id, [child]name, [mother]).
Fill in the gap in the SQL Statement to
select the [name]s of the [child]ren
and the corresponding [mother].
SELECT [child]name, [mother] FROM
[child] WHERE _____;

Asking to formulate an SQL Statement would address a higher level of problem-solving skills in this example. Therefore, an open (free-text) question type is more appropriate for testing programming problem-solving and programming skills. However, the description of scenarios with domain parameters increases the complexity of the question templates to ensure the readability of the resulting item templates. For instance, the domain parameter [child] occurs in singular, plural, and a combined variation. Furthermore, conditions such as 'name starts with the letter L' must apply to all domain models.

The generation of solution hints has to consider different problem-solution approaches. Student solutions to this example question may contain aliases, permuted attribute lists, or alternative condition formulation, such as:

```
SELECT c.childname, m.name
FROM child c NATURAL JOIN mother m
```

```
SELECT childname, mother
FROM child
WHERE mother IN (SELECT id FROM mother)
```

However, using keyword lists was insufficient for a fully automated evaluation of student solutions. For this example, a parser for SQL could be used to automatically evaluate the correctness of student solutions, as the author proposed (Saatz 2017).

3.4 Example 3: Programming Item

Item generation for imperative or object-oriented programming tasks is even more challenging as the solution requires a sequence of decisions. The following generated item, taken from an examination, measures the skills in programming in imperative (PL/SQL) programming languages.

Consider the following relation:

```
[enrolment] ([modulid],
[studentnumber], [status],
[enrolmentdate])
```

It should be ensured that a [enrolment] to a [module] can only be [deleted] if the [status] is "[enrolled]" and the [deletion] is done within [14 days] after the [enrolment date].

Implement this rule in Oracle and provide test cases to validate it. Pay attention to structured programming. In case of failure, meaningful error messages should be displayed.

In contrast to the second example task, this example question describes a real-world scenario calling for implementing a rule in an imperative database program. However, the question does not state what kind of database program (stored function, stored procedure or trigger) has to be implemented or how the implemented program has to be tested. To test the database program, the students must formulate data manipulation statements. Further constraints, for example, structured programming and meaningful error messages, are added to the question text to ensure similar difficulty levels for evaluation purposes. Although solution hints for such programming tasks could be created according to the domain model, the generated solutions do not cover all possible structural solution variations.

4 FIRST EXPERIENCES

Since 2021, the author has used the two-step workflow for the automatic generation of items with varying domain models. The generated items have been used in formative assessments of a database course since 2021 at the author's university. In each year, 148+/-6 students participate in the formal assessment. In the first examination in 2021, question templates were used, each covering sequenced tasks in declarative (SQL) and imperative (PL/SQL) programming languages. In 2021, 5 question templates covering a whole examination were used together with 18 domain models. Thus, 90 assessment

items were assigned individually to the participants in an open-book examination. Each question template contained problem-solving tasks, such as implementing a database scheme, queries, and views and implementing and testing a database program. However, solutions to generated items according to at least two question models seemed to have been exchanged between the participants during the examination due to similarities in the student solutions.

Therefore, since 2022, the learning management system has been used to assign randomly generated test items from 13 item pools to the students. In this examination, modelling items comprise the creation of installation scripts with more constraints for up to three relations, structurally comparable to example 1. The items testing the skills in declarative programming have had a higher difficulty level than provided by example 2 due to more complex solutions containing join and set operations, nested queries, or the definition of views. The imperative programming items are comparable to the given example 3 above. Due to the limitations of the Learning Management System, assigning each student individual tasks as intended was impossible. Therefore, 18 domain models were combined with six up to nine question templates for each task assigned randomly to the examinees. Thus, each participant got items according to varying domain models in this examination. The assessments comprised up to 35% of closed questions (single- and multiple-choice) and open free-text questions problem-based questions. However, using generated assignment tasks with individual assigned domain models does not prevent students from exchanging solutions during the online examination. Two malpractice cases occurred where students submitted the exact solution to open questions corresponding to another domain model or template. Typical student errors could explain some other instances of similarities between solutions. Differences in the difficulty level did not appear, apart from two cases of readability issues due to grammatical flaws in the reading text.

The assurance of the item template's quality is essential for successfully generating items using the proposed two-step workflow. Overall, reviewing the question templates is more demanding than reading plain text, as descriptions in question templates are more abstract. Therefore, an expert-based evaluation has been carried out on the test corpora instead of a student-based or model-based evaluation. The model-based and the student-based evaluation methods are not applicable due to the absence of a mathematical model when varying the domain context. In this

expert-based evaluation, the templates and the generated documents must be read carefully to spot any semantic, syntax, or language errors. After that, the expert must decide whether to change one of the parameter lists (semantic), the templates (semantic, syntax, language), or the corresponding word conjugation list (language).

5 DISCUSSION

Overall, the detection rate seems lower than in online-supervised examinations performed simultaneously at the author's faculty. Therefore, randomly assigning automatically generated items with varying domain models indicates a way of keeping students from exchanging solutions in online assessment.

The usage of automatic item generation in educational assessments promises reduced cost, time and effort in creating items for large item pools (Circi, Hicks, Sikali, 2023). This central promise of rapid item development for large item pools could be proven by applying the proposed two-step workflow for the automatic generation of items. Four challenges according to automatic item generation have been identified (Setianwan, Hidayah, and Kusumawardami, 2022):

- Generate various types of question types.
- Handle long and complex sentences.
- Dataset availability for non-English languages.
- Capability to control question difficulty.

The proposed two-step workflow contributes to all these challenges. Item templates were developed for single- and multiple-choice, fill-in-the-gaps, and free-text questions with or without integrated images. Extending to further question types or changing the learning management system is possible by developing new item templates. An item template is developed by parametrising a corresponding QTI file.

The development of further domain models requires the addition of domain parameters and their corresponding German language parameters. The language parameters are used to generate long and complex sentences to describe question scenarios and contexts based on different domain models.

Through artificial intelligence-powered tools, evaluating the generated test corpora regarding syntactical and language errors and even translating them to other languages might be possible. However, reviewing templates, domain-related parameters, and semantic correctness might remain expert-based, as understanding the structures, learning goals, and real-world contexts is necessary to check the correctness and readability.

The advantage of using question templates combined with various domain models is that it eases the effort of creating examinations of similar difficulty levels. Therefore, the question template determines the item's difficulty level. However, the used domain model might also influence the item's difficulty level. Chen et al. examined the effect of randomly chosen numerical parameters on the difficulty of generated questions (Chen et al., 2019). Their findings indicate only a limited impact of the parameters on the difficulty level in 5% of the examined AIGs. However, parameters defining domain models might have a more significant effect, as the students might be more accustomed to one domain model than another, which influences the readability of the task descriptions. Therefore, the influences of the parameter choices on the difficulty level of the generated tasks have to be examined further. However, constructing a similarity measure for automatically generated problem-solving tasks with different domain models is an area of interest in further research. Providing all structural solutions of open problem-solving tasks, such as programming tasks, for an automatically generated question item is even more challenging.

6 CONCLUSION AND OUTLOOK

A two-step workflow for automatic item generation can provide items with domain model variations for problem-solving exercises in computer science assessments. The generated corpora are useable in formative assessments in computer science education, providing individual assignment tasks to the students. To maintain and extend the corpora, additional templates and domain models should be developed for each examination period in the future.

One drawback of the proposed two-step workflow is that the information used to generate an item is scattered throughout various resources. Therefore, a database application should be used to maintain items, templates, and parameters more efficiently. Furthermore, the influence of domain models on the item's difficulty level has to be explored in more detail. Even more challenging is generating feedback for problem-based items, such as programming tasks, with various correct solutions. One possible solution might be integrating an artificial intelligence-powered tool in the proposed workflow to ensure the generated items' readability and to generate more specific feedback according to typical student errors.

ACKNOWLEDGEMENT

The author thanks Melanie Beutel for assistance with developing item templates for the learning management system ILIAS.

2015/ABP2017_paper_06.pdf (Last Accessed: 20.03.2024).

Xiao, W., Wang, M., Zhang, C., Tan, Y., Chen, Z. (2018). Automatic Generation of Multiple-Choice Items for Propositions Based on Word2vec. In *ICPCSEE 2018, CCIS 902*, pp. 81-95. Springer Nature Singapore Pte Ltd.

REFERENCES

- Anderson, L.W., Krathwohl, D.R. (2001). A Taxonomy for Learning, Teaching, and Assessing: a Revision of Bloom's Taxonomy of Educational Objectives. Longman, New York.
- Circi, R., Hicks, J., Sikali, E. (2023). Automatic item generation: foundations and machine learning-based approaches for assessments. In *Frontiers in Education*. 8:858273. doi: 10.3389/educ.2023.858273
- Chen, B., Zilles, C., West, M., Bretl, T. (2019). Effect of Discrete and Continuous Parameter Variation on Difficulty in Automatic Item Generation. In *AIED 2019*, LNAI 11625, pp. 71-83. Springer Nature Switzerland AG.
- D. J. Ferreira, A. P. Ambrósio, T. Nogueira, M. R. D. Ullmann and T. F. N. Melo, "Students' Perceptions of Applying Real-world Problem Solving in Computer Science Education : Case Study in Interaction Design," *2018 IEEE Frontiers in Education Conference (FIE)*, San Jose, CA, USA, 2018, pp. 1-8, doi: 10.1109/FIE.2018.8658458
- Foulonneau, M., Ras, E. (2013). Using Educational Domain Models for Automatic Item Generation Beyond Factual Knowledge Assessment. In *EC-TEL 2013, LNCS 8095*, pp. 442-447. Springer-Verlag Berlin Heidelberg.
- Gierl, M. J., Lai, H., & Tanygin, V. (2021). *Advanced methods in automatic item generation*. Taylor & Francis eBooks. Routledge. <https://doi.org/10.4324/9781003025634>
- Gierl, M. J., Matovinovic, D., Lai, H. (2019). Creating Content for Educational Testing Using a Workflow That Supports Automatic Item Generation. In *EAI International Conference on Technology, Innovation, Entrepreneurship and Education*, Lecture Notes in Electrical Engineering 532. Springer Nature Switzerland AG.
- Grammarly (2024). Grammar checker. Available at: www.grammarly.com/grammar-check (Last Accessed: 28.02.2024).
- Setianwan, H., Hidayah, I., Kusumawardani, S. (2022). Automatic Item Generation with Reading Passages: A Systematic Literatur Review. In: *8th International Conference on Education and Technology (ICET)*. IEEE.
- Satz, I. (2017). Wo steckt nur der Fehler in der SQL-Anfrage? Semantische Prüfung von Lösungen. 3. *Workshop Automatische Bewertung von Programmieraufgaben (ABP 2017)*, Potsdam, Germany. Available at: <https://ceur-ws.org/Vol->