A Structured (Java) Exercise Repository with Automated Feedback (SERF)

Lex Bijlsma
Open Universiteit
Faculty of Science, Department of Computer Science
6401 DL Heerlen, The Netherlands

Cornelis Huizing
Eindhoven University of Technology
Faculty of Mathematics and Computer Science
5600 MB Eindhoven, The Netherlands

Arjan Kok
Open Universiteit
Faculty of Science, Department of Computer Science
6401 DL Heerlen, The Netherlands

Ruurd Kuiper
Eindhoven University of Technology
Faculty of Mathematics and Computer Science
5600 MB Eindhoven, The Netherlands

Harrie Passier
Open Universiteit
Faculty of Science, Department of Computer Science
6401 DL Heerlen, The Netherlands

Erik Scheffers
Eindhoven University of Technology
Faculty of Mathematics and Computer Science
5600 MB Eindhoven, The Netherlands

Stefano Schivo
Open Universiteit
Faculty of Science, Department of Computer Science
6401 DL Heerlen, The Netherlands

Tanja Vos
Open Universiteit
Faculty of Science, Department of Computer Science
6401 DL Heerlen, The Netherlands

ACM Reference Format:

Lex Bijlsma, Cornelis Huizing, Arjan Kok, Ruurd Kuiper, Harrie Passier, Erik Scheffers, Stefano Schivo, and Tanja Vos. 2021. A Structured (Java) Exercise Repository with Automated Feedback (SERF). In *The 10th Computer Science Education Research Conference (CSERC '21), November 22–23, 2021, Virtual Event, Netherlands.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3507923.3507942

1 THE SERF REPOSITORY

SERF provides Java exercises to support training of (Java) OO programming skills. To make the repository teaching-approach independent (e.g., objects-first or objects-late), there is no approach-linked ordering or grouping of the exercises: SERF has a search function that enables to select individual exercises by training desire. Furthermore, to provide training in a manner that needs relatively little teacher support, e.g., in an on-line setting, solutions can be submitted to SERF, after which automated feedback is provided. A short description of the ideas is given below, more details can be found in the technical report [1].

CSERC '21, November 22–23, 2021, Virtual Event, Netherlands

ACM ISBN 978-1-4503-8576-3/21/11.
https://doi.org/10.1145/3507923.3507942

1.1 Tags

SERF has a repository, a database, of programming exercises. To make such a repository teaching approach independent, it should be possible to select an exercise on the basis of the current knowledge of a student rather than depending on to which stage in a specific course the student has progressed. More precisely, a student should be able to select the exercises that fit a training desire of that student.

Therefore, the training contribution of each exercise is made explicit. A *tag* is a word that characterises a (programming) knowledge item. The set of tags *Tag* for Java is the, carefully selected, set of words that identify programming knowledge items that pertain to Java: syntactic as well as semantic or conceptual knowledge items are used. The basic idea is, to use the tags to identify exercises that provide training in the corresponding knowledge item. So tags are added to exercises. To exploit this idea in a manner that enables efficient selection of appropriate exercises is a main contribution of SERF.

1.2 Knowledge graph and search function

Simply tagging each exercise with the training provided is impractical for two reasons. Firstly, an exercise may train very many different knowledge items, secondly it may require more prior knowledge than is feasible to indicate by tags directly. Both problems are solved by ordering the set *Tag* with an strong, *needs* and a weaker, *uses* prior knowledge relation between the knowledge items. This results in the first idea: a *knowledge graph* that records

the dependencies between knowledge items. The construction of the graph is crucial: general discussion as well as specific decisions for the Java domain can be found in [1].

The same notion of prior knowledge is used for the tagging of exercises. Selecting, using the graph, only top-tags in the orderings for tagging exercises keeps the tagging manageable.

The second idea is the implementation in the SERF tool of a search function that is developed and implemented as part of a tool that enables to find exercises that match a training desire. The searcher indicates, with tags as parameters, which knowledge is desired to be trained, and, with negative tags, which knowledge is to be avoided as yet-beyond the searchers capabilities. The search function, making use of the relations between tags as provided by the knowledge graph, then returns a set of exercises, with information about the required prerequisite knowledge for each exercise. The searcher can then, based on this information, refine the search and thus, iteratively, find a set of exercises that fits the training desire.

Note, that now putting an exercise in the repository requires tagging it! The submitter of the exercise has to provide the tags. Again the knowledge graph is used to guide and support the submitter to choose appropriate tags, e.g., only using the aforementioned top-tags for tagging the exercise,

1.3 Student submission and assessment

The SERF tool enables that solutions of exercises, i.e., programs, can be submitted. JUnit tests are performed by the tool on the solution, generating feedback. The tool provides the results in a suitable feedback format. Resubmission is possible, thus enabling to iteratively improve the solution.

Note, that now putting an exercise in the repository requires providing tests/feedback for it! There is a loosely defined standard format for exercises: exercise description and example i/o. To enable automated feedback on solutions, the precise input and output format for each exercise is explicitly given. There is a protocol for adding exercises to the repository with a review component that ensures the quality of exercises.

1.4 Steps in finding an exercise

To give an idea of the use of SERF, we indicate the steps in finding an exercise.

Primary actor: either teacher or student.

Step 1 Choose one or more tags that indicate the subject for which the exercise is to provide a practice opportunity.
Step 2 Indicate tags for related subjects that have not been mastered and hence should not be required for the exercise. For instance, a query might have search tag *repetition* and negative tag *array*. This step is optional, but it will reduce the work in the next steps.
Step 3 The system retrieves a set of exercise titles whose tag set contains at least one of the search tags of Step 1.
Step 4 The system removes exercises from the set where the needs tag set contains any negative tags listed in Step 2.
Step 5 The system displays a list of exercises together with their prior knowledge needs and uses tags.
Step 6 Repeat from Step 2, using a larger set of blocked knowledge items, until exercises are found that have no unwanted prior knowledge.

2 EVALUATION OF THE TOOL

The tool has been used at several institutions: OU, TU/e and NHL Stenden.

The results have been evaluated, a technical report is in preparation. The evaluation consists of quantitative analysis of logged information like submission behavior of the students for particular exercises, and on qualitative information obtained through interviews.

Some observations are the following. The search function was used, but, likely due to the small number of exercises currently in the database, also just browsing through these exercises occurred. Feedback, as intended, triggered resubmission. Compared to other on-line exercise databases, students positively valued the presence of more complex, advanced exercises than provided there - the tagging appears to be able to accommodate also these exercises.

3 DISCUSSION

The SERF tool enables to select Java exercises by training aim and provides feedback on solutions.

The two main ideas for selection are to use a prior knowledge graph to manage the complexity of knowledge item tags and to use a search function that, based on the knowledge graph, selects exercises that provide the desired training.

The main idea for feedback is that exercise-specific tests are automatically applied to submitted solutions and that information about the test results is then provided to the submitter.

These ideas can be used for any domain for which training of knowledge through exercises applies. The approach is beneficial in situations where several orderings of exercises depending on, e.g., teaching approach or interests, occur, and where there are extensive dependencies between knowledge items. To obtain the graph requires effort. If the domain is close to Java, say Python, much can be re-used from our graph. If the domain is quite different, still many of our techniques to obtain the graph can be re-used. The search function can be re-used if the relations in the graph are as in our graph: *needs* and *uses*; if different, adaptations should be made accordingly. Feedback on solutions is likely to be application domain dependent.

REFERENCES

[1] Lex Bijlsma, Cornelis Huizing, Arjan Kok, Ruurd Kuiper, Harrie Passier, Erik Scheffers, Stefano Schivo, and Tanja Vos. 2021. *Construction of a knowledge graph for exercise selection.* Technical Report. Open Universiteit.