XVII International Scientific Conference on Industrial Systems (IS'17)
Novi Sad, Serbia, October 4. – 6. 2017.
University of Novi Sad, Faculty of Technical Sciences,
Department for Industrial Engineering and Management
Available online at http://www.iim.ftn.uns.ac.rs/is17

System for automatic creation of equivalent task variations in the student tests

Milana Novković
(Teaching Associate, University of Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovica 6, 21000 Novi Sad, Serbia, mnovkovic@uns.ac.rs)

Srđan Sladojević
(Assistant Professor, University of Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovica 6, 21000 Novi Sad, Serbia, sladojevic@uns.ac.rs)

Abstract

Modern educators commonly use tests as a form of knowledge evaluation. However, they face a problem of creating multiple variations of a single test. In order to keep the balance between distribution and difficulty of tasks and also maintain the basic test structure and equivalent level of expected knowledge per variation, educators face a time-consuming challenge.

Specialized Java template based engine was used in order to minimize time consumption and simplify the process of test creation. The system enabled the creation of reusable templates as a basis on which test tasks were formed. Reusability of templates and variation of certain template parts as functionalities of this system provided test tasks that were slightly modified in different test variations. In this paper, equivalence of task distribution, difficulty and time consumption in the process of creating test variations was reviewed. The whole process was based on test cloning and enabling equality in task distribution by using same based templates to design diverse but equivalent tasks in test variations.

Key words: *Test variations, Java template engine, equivalence of test tasks, programming*

1. INTRODUCTION

Test as a form of knowledge evaluation is the most widely spread form of assessment in higher education. Traditionally, tests are bounded to specific subject or area. Here, the process of assessment of students knowledge in the area of programming in C# language is addressed. The process of testing from the teacher's point of view, difficulties in testing programming knowledge, the complexity of tests and time consumption during test generation process are observed in the paper.

In order to test programming knowledge, tests consisting of tasks in a form of small code examples that implement basic conceptual knowledge and operations are used.

During the development of this kind of tests, teachers face many challenges. Developed tests must exemplify valid assessment measures that correspond to the defined testing time frame. Besides complying to the language syntax and semantics, basic concepts must be implemented in such tasks in a manner they represent good examples of using language features on an expected level of difficulty. Furthermore, the teachers must be confident that the results of the tests are valid and can rely on the fact that unallowed actions like cheating did not contribute to the false representation of the results. Having that in mind existing problems grow when in order to minimize such actions teachers introduce multiple variations of a single test. Not only they need to come up with multiple test variations, they also need to keep basic structure, the same level of difficulty and expected knowledge among different variations. Distribution of tasks becomes an issue too. Tasks must be distributed equally but in the same time, they should be structurally aligned with slight differences. While trying to keep up with all of the mentioned necessities time consumption stands out as the most prominent problem. In order to get over these problems, a code generation system, which is able to generate tests in the area of programming is developed.

A new approach, test generation based on templates that can be randomized and used an unlimited number of times is introduced. Randomization applies to replacing certain parts of templates with random values. Templates at the end of the process act as small pieces of code that stand for test tasks. Generation of templates and their randomization is based on a Java template engine incorporated into the developed system. The system also enables test cloning, editing and provides different variations of the same test task in a manner that enables the teacher to maintain basic test structure and level of difficulty while complying to the language syntax and semantics ultimately saving time in the process of test creation.

The rest of the paper is organized as follows: Section 2 deals with the related published work. Implemented technologies and an approach used in system development are described in Section 3. Section 4 deals with discussion referring to system contributions to the process of test generation and lastly section 5 addresses our conclusions.

2. LITERATURE REVIEW

Integrated and structured knowledge is a basis for a deep level of understanding in any particular subject or content area. In order to measure the level of understanding of some particular subject or area, educators in any level of education use different knowledge evaluation methods. In higher education assessment of students knowledge can be done using different methods such as exams, essays, oral presentations, group project and etc. Those methods are used as a meaning to verify outcomes of the learning process [1]. Empirical evidence provided by Miller and Parlett in [2] indicates that student behavior and learning process are also influenced in great measure by assessment.

Fairness of assessment process is pointed out as the key aspect of assessment and is brought into contact with validity and reliability of assessment process. As achievement tests are still the significant part of the assessment process and in some cases main methods of assessment the aspect of fairness in testing is in question when multiple variations of the single test are given [1]. All thing considered, educators face a challenge and therefore really on the process of standardization, Ie standardized tests. Standardized achievement tests bring difficulty in terms of choosing test items that correspond to defined testing times, are good measures of particular content area and comply with the level of expected knowledge and skills in the domain [3]. Adding that to a need to produce multiple variations of a single test leads to increased time consumption that educators face in process of developing such tests. Also, the balance between task distribution and difficulty must be kept throughout all variations in order to achieve fairness. This paper addresses the matter of testing students programming knowledge through using standardized achievement tests.

The process of acquiring programming knowledge is highly complex and requires the variety of cognitive activities, understanding, conceptual knowledge and the structuring of basic operations like conditional statements, loops and etc [4].

Rogalski and Samurçay in [4] address the problem teachers are facing during learning beginners to programming by implementing the special framework in order to understand why certain programming concepts and procedures are difficult to learn and teach. Linn and Dalbey in [5] describe the ideal chain of learning programming by first introducing language syntax and semantics, then combine language features with required programming skills to develop programs and finally develop problem-solving skills.

Winslow in [6] points out that the key aspect of learning to program is in practicing and adopting basic facts, features, and rules which enable problem-solving in specified areas. He encourages starting with simpler problems, practicing and aiming to more complicated ones with reference to that it is necessary to understand the essence of the problem. In other words, students must have a good knowledge of language syntax and basic programming concepts as a basis to better problem-solving.

In order to test students programming knowledge, this paper refers to tests consisting of small pieces of code implementing basic operations, facts and programming rules. That small piece of code serves as test tasks where students are being questioned what would be the output of that piece of code if it would be executed in the specified environment. In order to answer, students must have the good understanding of basic programming concepts and rules.

In [7] and [8] Pathak, Brusilovsky and Sosnovsky address the problem of the same set of questions usage for the whole class. They address the cheating problem as a problem that prevents teachers to rely on the results of the test. In order to test students in a way that cheating is minimized, multiple variations of a single test were used. A small piece of code that represents task of the test is slightly altered and spread across multiple variations of a single test. Pathak and Brusilovsky also developed a system that allows teachers to create parameterized questions for the quiz. Quiz relies on the teachers to provide the core content of the question which represents parametrized fragment of code. Our system runs on the same principle. Teachers provide the core template which acts as the basis of test task with parts that can be randomly changed by the system. Essentially it is a parametrized piece of code that is created by the teacher and randomly filled with different parameter values. By that, the main structure of the task is maintained but modified in certain parts by the system. Once the teacher provides the template, that same template can be filled with randomized values and used multiple times, every time producing different output.

Brusilovksi and Sosnovsky in [8] describe a system for generating parameterized exercises for C language that automatically evaluates given student answers. The system by the name QuizPack enables web-based quizzes through which teachers can evaluate student knowledge, but students also can improve their knowledge of semantics and overall programming skills. The system is also based on templates with parametrized pieces, where one template can produce a large number of different questions [8]. In our case system can randomize values of given parameters as many times it is necessary thus also giving a lot of different questions based on the same structure provided by a single template. Different template outputs brought from one template represent variations of one test task through multiple test variations. One created template can be used not only for one course or test, but multiple courses and tests thus decreasing the overall time needed for creating multiple tests and their variations.

3. MATERIALS AND METHODS

The implemented technologies and capabilities of the developed system are described further in detail. The focus is being set on the process of template creation and it benefits.

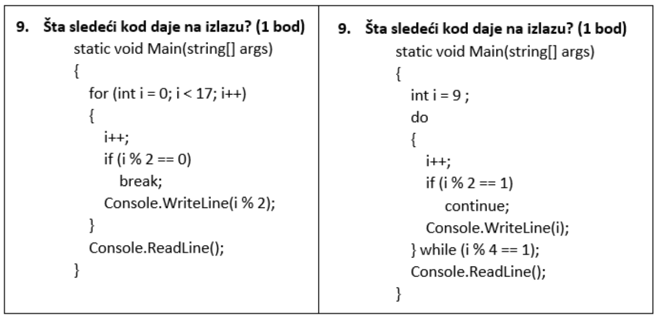
3.1. **Implemented technologies**

The system described in this paper represents a Java application which is thoroughly based on the REST (Representational State Transfer) which is an architectural style with lightweight communications between the producer and consumer [9][10]. RESTful web service provides resources and clients to exchange resource representations as parts of their interactions [9][11]. Since the system is web-based this service enables sending HTTP requests to Web pages in order to obtain the specified resources [9].

Part of the system in charge of generating test tasks in form of templates is based on an Apache FreeMarker template engine [12]. Freemaker is a free template engine in a form of a Java library, which provides powerful syntax for generation of templates. This template engine is an open-source software that enables programmers to embed the source code into different products [13]. Template engine operates on the specific data model which enables generation of textual templates that can output different representations of data in a form of HTML, configuration files, source code, etc [12][14]. The template engine is employed with the purpose of creating templates, which would eventually produce the source code. Due to its readability and adaptability, this template engine is used for many purposes. In [15] FreeMaker template engine is used as a basis for constructing the parser for the kernel of the compiler subsystem. It purpose varies from generating Software Factory for pervasive systems over rendering HTTP replies of emails to modeling and implementation of catalog cards [13][15][16]. The system described in this paper essentially enables code generation driven by the described template engine. Apache FreeMaker template engine brings many advantages into the system that incorporates its functionalities. It provides general-purpose templates much faster than other similar tools and enables templates to be loaded and reloaded during runtime without the need to redeploy the application [17].

3.2. **Code generation system: The approach and the system**

The process of developing the code generation system began with analyzing previously existing tests related to the programming area in order to identify basic tests and task structure. Existing tests consisted of a random number of tasks, each of which contained five or more lines of code. Student should give the answer referring to the output (which would be generated if that piece of code would be executed). Two of those tasks are presented in Figure 1.



9. Šta sledeći kod daje na izlazu? (1 bod)	**9. Šta sledeći kod daje na izlazu? (1 bod)**

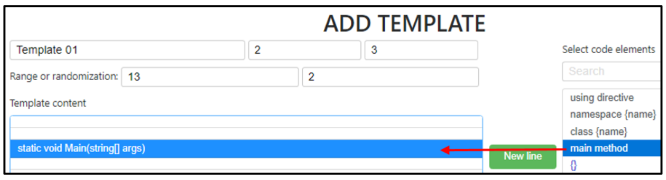
```
static void Main(string[] args)
{
    for (int i = 0; i < 17; i++)
    {
        i++;
        if (i % 2 == 0)
            break;
        Console.WriteLine(i % 2);
    }
    Console.ReadLine();
}
```

```
static void Main(string[] args)
{
    int i = 9 ;
    do
    {
        i++;
        if (i % 2 == 1)
            continue;
        Console.WriteLine(i);
    } while (i % 4 == 1);
    Console.ReadLine();
}
```

Figure 1. Example of test tasks written in C# programming language

The challenge was to design a system that can work for a large variety of programming questions wrapped in the form of small pieces of code. The system is based on the idea that each task that it produces should maintain basic structure, but differ in some parts that do not affect the structure but only the solution of the task.

Developed system has a user interface which enables teachers to create the desired template by selecting options from the list and placing them in specified locations. Each option from the list is a piece of the template, that reflects some basic programming concept, for example, variable, any type of loop, collection, array, etc. Figure 2 displays one part of the user interface intended to be used for template creation.



ADD TEMPLATE

Template 01	2	3
Range or randomization: 13	2	

Template content

```
static void Main(string[] args)
```

Select code elements

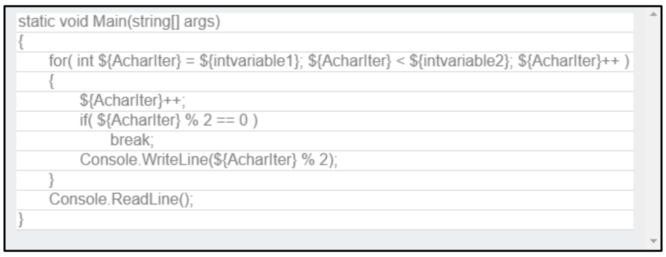
- using directive
- namespace {name}
- class {name}
- main method
- {}

Figure 2. User interface for creating templates

The user first selects the desired row from the list on the left. Selected row or line on the left side represents the place where the selected template item from right side list would be placed.

Every template has a specified name by which it can be identified and the number of points that task created with that template would bring in the test.

In order to facilitate and provide reusability of created templates system enables placement of created templates into MySql database. Template content, name and number of points it brings are stored in the database. This feature enables reuse and editing of templates. One such template created by the described user interface is displayed in Figure 3. The displayed template is a template generated on the basis of the task presented on the left side in Figure 1 displayed above in the text.



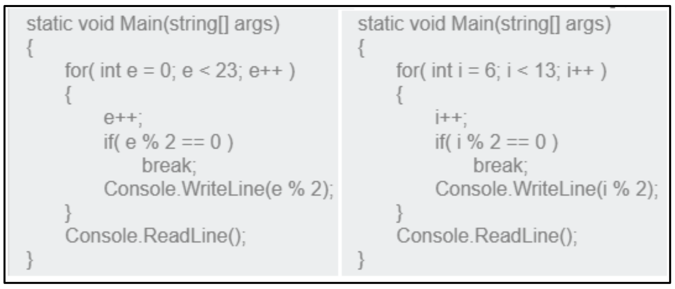
```
static void Main(string[] args)
{
    for( int ${AcharIter} = ${intvariable1}; ${AcharIter} < ${intvariable2}; ${AcharIter}++ )
    {
        ${AcharIter}++;
        if( ${AcharIter} % 2 == 0 )
            break;
        Console.WriteLine(${AcharIter} % 2);
    }
    Console.ReadLine();
}
```

Figure 3. Example of the template based on test task displayed on the left side in Figure 1

Once made, the template is permanently stored in the database and can be used multiple times, each time giving different variations of the task.

Motivation is to create a system that would allow teachers to create one template which would serve as a basis for multiple test tasks nor then just store fully developed tasks to the database. Hence, templates contain small pieces of the text that can be programmatically randomized and changed by the system. As seen in Figure 2 those pieces are placed inside curly brackets preceded by a dollar sign. Through multiple variations those values are changed but the basic structure of the task remains the same.

RESTfull services on which this system is based provide that functionality of filling the template with randomized values, but also storing the template into the database and editing the template. The teacher only provides the core of the task, and the system does the rest - randomization of a template. As a result of template randomization, the final tasks are generated. Generated tasks can be later used in process of test creation by simply selecting the task which would be placed into the test. Figure 4 display just some of the variations created by the template represented in Figure 3.



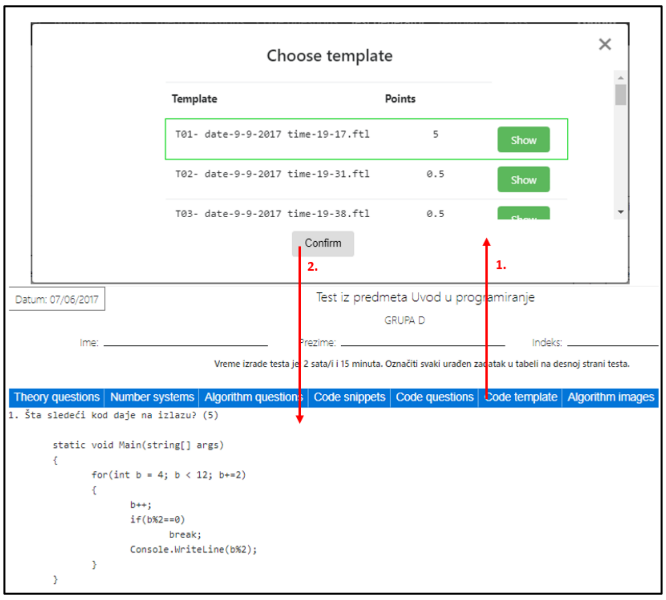
```
static void Main(string[] args)
{
    for( int e = 0; e < 23; e++ )
    {
        e++;
        if( e % 2 == 0 )
            break;
        Console.WriteLine(e % 2);
    }
    Console.ReadLine();
}
```

```
static void Main(string[] args)
{
    for( int i = 6; i < 13; i++ )
    {
        i++;
        if( i % 2 == 0 )
            break;
        Console.WriteLine(i % 2);
    }
    Console.ReadLine();
}
```

Figure 4. Example of task variations based on the template from Figure 2

The current version of the system supports programming related tasks based on C# programming language.

Besides mentioned capabilities, system also enables the teacher to incorporate created templates into tests. Figure 5 displays the process of incorporating created template into the test. The user selects the button with the title corresponding to template creation and in the newly opened window selects the desired template. Once the template is selected, and selection is confirmed, the system randomizes the values and displays the template as the test task as shown below.



Template	Points
T01- date-9-9-2017 time-19-17.ftl	5
T02- date-9-9-2017 time-19-31.ftl	0.5
T03- date-9-9-2017 time-19-38.ftl	0.5

Datum: 07/06/2017

Test iz predmeta Uvod u programiranje

GRUPA D

Ime: ______ Prezime: ______ Indeks: ______

Vreme izrade testa je 2 sata/i i 15 minuta. Označiti svaki urađen zadatak u tabeli na desnoj strani testa.

```
1. Šta sledeći kod daje na izlazu? (5)

        static void Main(string[] args)
        {
                for(int b = 4; b < 12; b+=2)
                {
                        b++;
                        if(b%2==0)
                                break;
                        Console.WriteLine(b%2);
                }
        }
```

Figure 5. Process of test creation

Once the test is created it can be cloned. Cloning in this context refers to keeping the test structure and distribution of tasks, and since tasks are basically templates that are randomly filled by the system, the whole test can be replicated in a way that each individual task is just randomized again and placed in test variations.

Template creation as application feature is just a small part of the system. The system also incorporates a large database of different kinds of programming questions, starting from theory questions, across algorithms to questions referring to number systems. It also provides generated tests in a form of PDF documents that can be downloaded.

4. RESULTS AND DISCUSSION

Described system was originally developed to provide teachers an easier way of creating multiple test variations and consume less time in the process. It is started from the general idea to keep the balance between distribution and difficulty of test tasks. Teachers as creators of the template are the ones that determine the difficulty of the task, make sure that the task represents the appropriate measure for assessing knowledge in the subject area and corresponds to the area that is being tested. In order to create the test, the teacher has to provide templates that can be used as test tasks.

The process of template creation leads to less time consumption, which depends on the size and the number of templates. But on the other hand, teachers can use already generated templates, previously stored in the database, as test tasks, thus conserving time. Furthermore, every template can also be edited and with slight moderations provide a different task. In addition, once the test is created and all templates are placed as test tasks, it is only a matter of seconds to create multiple variations of that test.

The features of the system also enable teachers to create different kinds of templates and ultimately after inserting them to the test, download created test in a form of the PDF document. Furthermore, since the tasks are just moderate changes of the specified template, the level of the expected knowledge does not differ across multiple test variations. Thus, equal conditions of testing are provided together with the fact that teacher can rely more on the validity of the results.

In the process of template generation, the system enables teachers to generate lines of code without writing any of them. Teachers just select code blocks they want to implement as parts of the template.

Other advantage lies in the fact that the system is web-based and can be accessed from any location with the condition that the Internet connection is provided. The teacher can also input new questions related to other areas of programming and use them to generate tests. So, not only tests with templates can be created, but also different combinations of that questions and templates can be used as test tasks.

5. CONCLUSION

The system for automatic creation of equivalent task variations in the student tests including its development process is described in this paper.

System uses templates for tasks modeling. Generated templates increase the productivity of teachers enabling them not only to conserve time but produce a range of task variations with same structure and level of difficulty. The system also enables task generation without writing any lines of code, therefore takes care about language syntax and semantics minimizing the effort compared to writing code directly.

Test generation described in this paper collides with the aspect of fairness in the process of assessment where tests are equally structured, require the same level of expected knowledge per variation and maintain balanced task distribution among multiple test variations.

It is planned for the system to be expanded by adding some new features. By implementing the language interpreter, a system could interpret task and thus produce solutions or outputs of the tasks. As a result, teachers could not only generate tests but their solutions too.

One further step would be an expansion of the system in another direction. With an embedded interpreter, the system could be an excellent learning and practicing platform for the students. Not only they could extend their knowledge by practicing on multiple different examples, they could benefit from the randomization process by seeing how one piece of code behaves with slight changes and compare their answer with the answer provided by the system. Finally, it is planned for created system to be incorporated into the process of test creation for some subjects at the University and evaluate the time consumption in different situations whether the tests are created from the beginning, edited or cloned.

6. REFERENCES

[1] McDowell, L. (1995), "*The impact of innovative assessment on student learning*". Programmed Learning, *32*(4), pp.302-313.

[2] Miller, C.M. and Parlett, M. (1974), "*Up to the Mark: A Study of the Examination Game*".

[3] Popham, W.J. (1999), "*Why standardized tests don't measure educational quality*". Educational leadership, *56*, pp.8-16.

[4] Rogalski, J. and Samurçay, R. (1990), "*Acquisition of programming knowledge and skills".* Psychology of programming, *18*(1990), pp.157-174.

[5] Linn, M.C. and Dalbey, J. (1989), "*Cognitive consequences of programming instruction".* Studying the novice programmer, pp.57-81.

[6] Winslow, L.E. (1996), "*Programming pedagogy—a psychological overview*". ACM Sigcse Bulletin, *28*(3), pp.17-22.

[7] Pathak, Sharad, and Peter Brusilovsky. (2002), "*Assessing student programming knowledge with web-based dynamic parameterized quizzes*." *Proc. of ED-MEDIA.*

[8] Brusilovsky, P. and Pathak, S. (2002), "*Assessing student programming knowledge with web-based dynamic parameterized quizzes*". In EdMedia: World Conference on Educational Media and Technology (pp. 1548-1553). Association for the Advancement of Computing in Education (AACE).

[9] Vinoski, S. (2007), "*REST Eye for the SOA Guy*". IEEE Internet Computing, *11*(1).

[10] *"REST (representational state transfer)*", available at: http://searchmicroservices.techtarget.com/definition/REST-representational-state-transfer (accessed: 05 June 2017).

[11] Sheth, A.P., Gomadam, K. and Lathem, J. (2007), "*SA-REST: Semantically interoperable and easier-to-use services and mashups*". IEEE Internet Computing, *11*(6).

[12] "*What is Apache FreeMarker?"*, available at: http://freemarker.org/ (accessed: 03 Jun 2017).

[13] Radjenovic, J., Milosavljevic, B. and Surla, D. (2009), "*Modelling and implementation of catalogue cards using FreeMarker"*. Program, 43(1), pp.62-76.

[14] Muñoz, J. and Pelechano, V. (2006), "*Applying Software Factories to Pervasive Systems: A Platform Specific Framework*"*.* In ICEIS (3) (pp. 337-342).

[15] Van Weert, P., Schrijvers, T. and Demoen, B. (2005), "*KU Leuven JCHR: a user-friendly, flexible and efficient CHR system for Java*". *In Proceedings of the 2nd Workshop on Constraint Handling Rules* (pp. 47-62). Deptartment of Computer Science, KU Leuven.

[16] Rosenberg, F., Curbera, F., Duftler, M.J. and Khalaf, R. (2008), "*Composing restful services and collaborative workflows: A lightweight approach*". IEEE Internet Computing, 12(5).

[17] Benato, G.S., Affonso, F.J. and Nakagawa, E.Y., "*Infrastructure Based on Template Engines for Automatic Generation of Source Code for Self-adaptive Software Domain".*