Educational tests in “Programming” academic subject development

Maksimenkova Olga
National Research University Higher School of Economics
Moscow, Russia
e-mail: omaksimenkova@hse.ru

Vadim Podbelskiy
National Research University Higher School of Economics
Moscow, Russia
e-mail: vpodbelskiy@hse.ru

***Abstract*—Educational tests are quite popular as a type of learning outcomes checking in public education and in commercial education nowadays. This work is devoted to test for students, who study programming in Universities. Educational tests in “Programming” academic subject development and statistical analysis principles are described and accumulated here.**

Keywords: software engineering education,educational tests, tests, programming academic subject

INTRODUCTION

Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering recommends such courses as “Computer Science”, “Programming basics”, “Programming” etc. to be taught to train an undergraduate in Software Engineering. Nowadays in Russia syllabuses of subjects, which are widely connected with programming (for example, “Programming” or “Information Technologies”, etc.), obligatory contain a part about one or another programming language. The most popular programming languages: C++, C# and Java are taught all over the Universities.

The problem of checking theoretical knowledge of syntax and semantic of a programming language and practical skills is significant and quite complicated. Well-prepared educational test can help in resolving all these problems by checking all of them. It should be said that educational testing is widely applied practically in specialists’ certification by such firms as Microsoft, Cisco, 1C etc.

This work is devoted to common questions of educational tests statistical analysis and development for students, who study programming in Universities. The research which is described in this work is based on tests in C# programming language which are given to the students who study programming within the academic subject “Programming” which is contained in the discipline of the Software Engineering.

ESTIMATING OF LEARNING OUTCOMES

Desire learning outcomes of a process of learning are formulated by the academic staff, preferably involving student representatives in the process, on the basis of input of internal and external stakeholders. Competences are obtained or developed during the process of learning by the student.

Learning outcomes are statements of what a learner is expected to know, understand and/or be able to demonstrate after completion of learning.

Competences represent a dynamic combination of knowledge, understanding, skills and abilities. Fostering competences is the object of educational programmes. Competences will be formed in various course units and assessed at different stages [9].

Tests, which are used to estimate students training level in a current academic discipline, are criterion-referenced (or should be such). Their goal is to reveal an examinee’s level of required knowledge, abilities and skills. Thus, the main goal of testing is establishing minimum number of points, achievement of which is enough to give a student a good mark.

A level of dynamic constituent parts of competence measurement is a difficult to create an algorithm and ambiguous in checking works problem. It needs to describe conceptual models, which are different to present.

Nowadays three dimensions model is certified and widespread.

First part of this model is a content, which is provides a content validity of a tool set, or to be more precise its compliance with educational programmes. Second part of the model are the process requirements – a type of test questions in a general case. Third part of the model is a level of a cognitive activity [5].

COMMON PRINCIPLES OF TEST’S CONTENT SELECTION

Recommendations in educational tests in programming development are given in this work. Collected common principles of tests in programming creation are based and correlated with common principles of content selection.

Content selection common principles provide high content validity support. Content validity is the estimate of how much a measure represents every single element of a construct [6, 12, 5].

1. Representativeness principle. This establishes a procedure of test content selection to provide optimum completeness and correctness of test content proportions.
2. Significance principle. This orders to put on the test the most significant content items, which are connected with the key topic of the course. The key topic extraction needs course content to be structured before its putting on the test.

3. System principle. This means that content items are put in order, and are connected with each other with a special hierarchy and a common knowledge structure. Following this principle test may be used not only to check educational achievements but to estimate knowledge structure of students' quality as well.

TYPES OF TEST QUESTIONS

Before describing tests in programming structure and separating questions types, which allowed estimating learning outcomes, we will give a brief test questions review. To make further narration more convenient we'll use symbols.

Test questions are divided into several types. Using different kinds of questions in a test can improve its quality and make it more flexible.

Question types:

1. Multiple choice questions (MCQs) – student should choose one from a list possible answers.
2. Multiple response questions (MRQs) – student should choose one or more from a list possible answers, one or more (even all) options can be the keys.
3. Text/Numerical question (Short answer questions – SAQ) – student should input text, numbers or both into a special empty text field.
4. Matching questions involve linking items in one list to items in a second list.
5. True/False questions require a student to assess whether a statement is true or not.
6. Author's type of questions [5, 6].
7. etc. [2, 5, 6]

Traditionally a MCQ or a MRQ consists of:

- a stem – question text;
- options – the choices are given after the stem.

The correct answer (or answers for MRQs) in the options list is called "the key". The incorrect (but verisimilar) answers in the options list are called "distracters" [2].

Example 1. MRQ question

System class String static methods which are returned a string are:

1) `Join()`
2) `Equals()`
3) `Copy()`
4) `CompareOrdinal()`
5) `Intern()`

Example 2. MCQ question

Compilation and running of this code:

```
int dif;
char ch1 = 'A';
char ch2 = 'c';
dif=Char.ToUpper(ch2).CompareTo(ch1);
Console.Write(dif);
```

stem

will output:

options

1) -1
2) 1
3) 2
4) 34
5) 0

distracters

key

Example 3. SAQ question

Compilation and running of this code:

```
int a = 7;
int i = 0;
while (a == 7)
{
      if (i < 4)
            Console.Write(i++);
      else
            Console.Write(++i);
      break;
}
```

will output: ___

TEST IN PROGRAMMING COMMON STRUCTURE

Test consists of a number of questions (test problems). Besides dividing test questions into listed classical question types it is desirable to use another one classification. Each test problem consists of question's body which can contain whole programs or code fragments.

Types and features of the test problems

A-type: Questions for checking a programming language syntax and semantic theoretical knowledge. This type of questions is represented by both the MCQs and MRQs. Questions of this type do not contain code of whole programs of all-in-one blocks of code.

B-type: Applied questions for checking practical skills; functionality skills analysis, and development programs according to a given functionality skills. This type of questions is represented by MCQs, MRQs or (the better one) SAQs. Questions of this type may contain program's code or all-in-one blocks of code.

B-type question's features

- Program functionality or all-in-one code block functionality analysis. Test problem is designed as MCQ. Student is asked to resolve what the code that is given in a stem execution result is. Source data are defined by the program or by a user. In the last case their values should be given in a stem. If a syntax error is intentionally added to the code the option is given (Example 2, 3).
- Analysis if a program meets requirement functionality. Test problem is designed as MCQ or MRQ. Student is provided with the whole description of a program or a code block purpose, which is given with gaps. One or more options

should provide requirement functionality if they are put in the gaps.

- Causes of departure from predefined behavior analysis. Test problem is designed as MCQ or MRQ. Stem contains a program (block of code) purpose description, a whole program or all-in-one code block and a result of its execution. Student finds out presence of deviation of the program's result from the predefinition. Student detects how far result of a program deviates from the predefinition one of defines changings which make program meeting requirement functionality.

Test questions which are connected with syntax errors in a program's code should be designed as MCQs or MRQs with concrete clear options. This means that the given options shouldn't contain compiler's messages. Questions like "Which message will be generated by a compiler as a result of compilation?" are also unacceptable.

Quantity of A- and B-type questions in a structure of a test should be balanced and can't be changed by a test-developer of in a concrete test. Topic's contents distribution by question's types is free and is defined by a test-developer.

Test decoration requirements

Test, for example, can be prepared as separate MS Word file and contain 30 - 40 items.

Questions should be formed as a table, left column for a number of question and keys. Key for a short-answer question is indicated as a value, for MCQs and MRQs numbers of right options are enumerated (see Table 1).

Each stem of MCQ or MRQ is followed by five options. For MRQs even all of the options can be keys.

A stem and the options shouldn't be more than 24 text-strings long.

TABLE 1. TEST QUESTIONS DECORATION (FRAGMENT)

9	**In the given code block which determines if a string length and digits sum in it are equal**

35	`string str = Console.ReadLine();`
	`int sum = 0;`
	`for(int i = 0; i < str.Length – 1; i++)`
	`if(str[i] > '0' – 1 && str[i] < '9' + 1)`
	`sum += str[i] – 0;`
	`Console.WriteLine(sum == str.Length);`
	programmer made mistakes:

	1) only 1..8 digits summs up 2) a string char is addressed by index 3) digits codes are summed up 4) variable `sum` is called in `for` 5) the last char of string isn't analyzed
10	**Mark commands after adding which into the following code block execution will output 6**

1345	`using System;`
	`class Program`
	`{`
	`static void Main()`
	`{`
	`// TODO add your code here`
	`}`
	`}`

	1) `Console.WriteLine(12 >> 1);` 2) `Console.WriteLine(2 & 4);` 3) `Console.WriteLine(5 ^ 3);` 4) `Console.WriteLine(6 | 4);` 5) `Console.WriteLine(7 >> 1 << 1);`
11	**Compilation and running of this code**

1:0:True	`string s1 = "Cat", s2 = "cat", s3 = "Cat";`
	`Console.Write(s1.CompareTo(s2)+":");`
	`Console.Write(s1.CompareTo(s3));`
	`Console.Write(":" + (s1 == s3));`
	will output:____

Tests which are developed according to given specification are suitable to be used in two cases:

- Paper-and-pencil testing (blanks);
- Computer testing.

The time to complete a test (ex. 30 - 40 questions) is limited (ex. from 40 to 60 minutes). Results are assigned using a binary scale (dichotomous appraisal plan).

SAQs of B-type can be redesigned into MCQs or MRQs for the computer testing needs.

THE RESEARCH

A group of 88 first-year students of Software engineering department was given an exam in a test-form after a first semester of the "Programming" academic subject. Test's questions were designed according to the structure was given above. Test consisted of 30 questions; time was limited by 30 minutes. This test we will call in short "the test" here and below. The test was given in a computer-based form.

Examinee's results were put in a table – response matrix. We used a dichotomous scale, for the items. Response matrix was used as a base in calculating primary scores and in visualizing the distribution of scores in a diagram (Fig. 1).

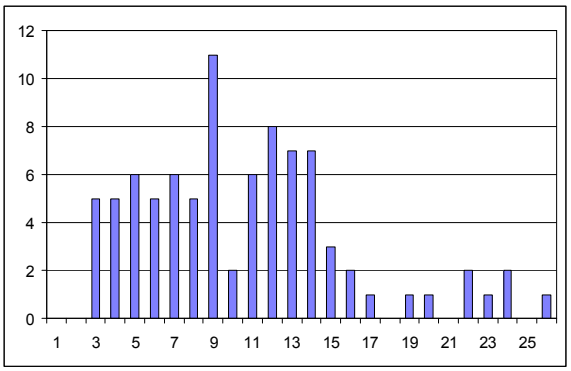

Figure 1

The mean is the average of all of the scores on the test. For the current test it was calculated as:

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N}$$

where x_i are the individual scores, N – a number of examinees.

There is a downfall in the middle of a diagram, near the mean ($\bar{x}$ = 10, 47). This means that we have got a bimodal deviation curve. Besides that we can draw a conclusion that all the examinees are divided into two level-groups. It seems to be quite reasonable because of peculiarity of the academic subject "Programming". It is learnt by the first-year undergraduates, former schoolchildren, whose base knowledge in programming is quite different. Some of them are lyceums-graduates and

well-trained in IT and programming, the others finished secondary schools and haven't got much special knowledge [3].

Item analysis statistics

The item difficulty (or item difficulty index) (*p*-value) is a measure of the proportion of examinees who answered the item correctly. The difficulty of an item *j* is calculated as:

$$p_j = \frac{N_j}{N}$$

where N_j – number of examinees with a score 1 on item *j*, and *N* – number of examinees [10].

TABLE 2. ITEM DIFFICULTY (FRAGMENT)

Item number	11	12	13	14	15	16	17
p_j	0,736	0,31	0,057	0,483	0,598	0,552	0,391

The middle *p*-value for our test is 0,353, minimum value is p_{13} = 0,057, maximum p_{11}=0,736. The test hasn't got very difficult and very easy items. On the whole, the test is corresponded to the examinees' level.

The item discrimination index is a measure of how well an item is able to distinguish between examinees who are knowledgeable and who are not [10]. High values of this index correspond to good items and low – to bad ones. The item discrimination index for item *j* is calculated as:

$$D_j = \frac{n_j^b}{N^b} - \frac{n_j^w}{N^w}$$

where n_j^b – number of examinees from the best group with a score 1 on item *j*, n_j^w – number of examinees from the worst group with a score 1 on item *j*, N^b – number of examinees in the best group, N^w – number of examinees in the worst group.

In short, item discrimination indexes which are calculated for the test show that it works quite well for this scope of students. All of item discrimination indexes are positive, so better examinees did test better then worse examinees. Items 9, 10, 13, 28, 30 have D_j < 0,19. Items 1 and 22 have 0,19 < D_j < 0,3. These items should be reviewed or altered.

The point-biserial correlation is the correlation between right (wrong) scores that students receive on a given item and the total scores that the examinees receive when summing up their scores across the remaining items [11]. The point-biserial correlation index is calculates as:

$$r_{pbjs} = \frac{(\mu_+^j - \mu_x)}{\sigma_x}\sqrt{p_j/q_j}$$

where μ_+^j – the average total score for those students who answered item *j* correctly, μ_x – the average total score for the whole group of examinees, σ_x – the standard deviation of the total score for the whole group of examinees, p_j – the difficulty index for item *j*, $q_j = 1 - p_j$ [12].

The point-biserial correlation indexes for the all test's items are positive (Table 2). High negative value means that examinees who scores well on the test have a lower probability of answering this item correctly.

TABLE 3.THE POINT-BISERIAL CORRELATION

Item number	1	2	3	4	5	6
r_{pbjs}	0,405	0,286	0,223	0,34	0,269	0,585
Item number	**7**	**8**	**9**	**10**	**11**	**12**
r_{pbjs}	0,448	0,333	0,038	0,225	0,263	0,473
Item number	**13**	**14**	**15**	**16**	**17**	**18**
r_{pbjs}	0,383	0,549	0,409	0,385	0,575	0,366
Item number	**19**	**20**	**21**	**22**	**23**	**24**
r_{pbjs}	0,596	0,572	0,619	0,324	0,456	0,526
Item number	**25**	**26**	**27**	**28**	**29**	**30**
r_{pbjs}	0,572	0,562	0,374	0,29	0,539	0,282

The retest for the same examinees on this test is impossible. Thus the split-half method or the KR-20 may be used in estimating reliability of the test.

The KR-20 (Kuder-Richardson 20) was developed to handle only dichotomously scored items [12]. So far as we used dichotomous scale for the items of the test we apply the KR-20:

$$r_{KR-20} = \frac{N}{N-1}\left(1 - \frac{\sum_{j=1}^{N} p_j q_j}{D_x}\right)$$

where *N* – number of test items, p_j – the difficulty index for item *j*, $q_j = 1 - p_j$, D_x – the total test variance.

For the test $r_{KR\text{-}20}$ = 0,826.

Using the standard deviation of the total score (σ_x= 5,38) and the reliability of the test we can calculate σ_E – standard error of measurement (SEM) to estimate how close to the true score the obtained score is [12].

$$\sigma_E = \sigma_x\sqrt{1 - r_{KR-20}} = 2{,}242$$

Prediction interval with the 5% level is:

$$(x_i - 2\sigma_E, x_i + 2\sigma_E) = (x_i - 4{,}484, x_i + 4{,}484)$$

According to the test theory and practice such reliability is quite acceptable and the test can be considered professional enough.

THE RESULTS SCALING

It will be recalled that the main goal of testing is establishing minimum number of points, achievement of which is enough to give a student a good mark.

The lifetime of study programs of the modern academic subjects is quite short. Besides that, groups of examined students are small. Usually they count one or two students' groups (25-30 persons in each), four-five groups are infrequent occurrence. Thus, there is impossible to get regulations which widely demonstrate a test quality and global scores of examinees, because of absence a huge representative sampling of examinees in the University.

It is also almost impossible to organize a peer review of all the tests, because a number of lecturers who are familiar with a content of each current test is limited and this kind of activity isn't traditionally contained into their syllabuses. The same troubles take place in setting a minimum number of points using a posteriori examination of a test and the results of it.

So, in spite of requirement of criterion-referenced type of interpretation of the test, we have to resort to norm-referenced type of interpretation. This gives a chance to determine an examinee's relative position within the specified group, to rank examinees according to their scores and to estimate indirectly difficulty of a test.

Reverting to the test, we used two approaches to interpret individual scores of the examinees.

First of all, we reduced individual scores to a standard Z-score. A mark of an examinee is calculated as:

$$Z_i = \frac{(x_i - \mu_x)}{\sigma_x}$$

where x_i – an individual score, – the average total score for the whole group of examinees, – the standard deviation of the total score for the whole group of examinees.

z-values lies between -3 and +3, so they can't be directly use as marks in the ten-point score [3]. To transformation from z-values to ten-point values we use the following ratio:

$$B_i = \frac{10x_i(Z_{max} - Z_{min})}{(Z_i - Z_{min})}$$

where x_i – an individual score, Z_i – a mark in z-score, Z_{max} – maximum mark in z-score, Z_{min} – minimum mark in z-score.

For the test Z_{min} = -1,95, Z_{max} = 2,89.

The threshold value for the ten-point score is 3,5. Marks less than this value are unsatisfactory, marks from 3,5 to 4,5 are satisfactory, from 4,5 to 7,5 are good, more than 7,5 – excellent [3].

To compare ten-point score with the traditional scores, we made a transformation from z-values to t-score:

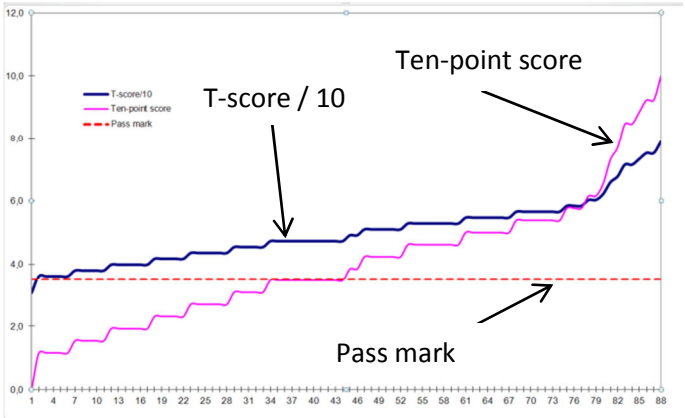

Figure 2. Total scores

$$T_i = 50 + 10Z_i$$

Limit values of t-score marks are 30,54 and 78,89. To make comparison easier results of calculating is put into a graph which is given on Figure 2. T-score values on Figure 2 were decreased in ten times.

CONCLUSION

Computer testing in "Programming" and "Algorithmic languages" is an effective method of impartial assessment of student's achievement at the different stages of educational process. Testing is reasonable in preliminary examination to find out groups of students whose further education needs additional consultations or adaptation courses. Testing can provide self-control in learning academic disciplines which are connected with programming. Testing is effective as intermediate test-check and very useful then using as a part of total test-check, exam for example.

It should be noted that some factors prevent from regular and multi-faced usage of computer testing in institutions of higher education.

- Test development is a high work content activity, which isn't taken in consideration in lecturers' syllabuses.
- Occurring everywhere absence of licensing computer testing software with a special functionality to collect individual scores of examinees and items.
- Neediness of tests' approbation. Primary contingent of examinees is absent (tests are prepared to small groups of students; they can't be approved before their application). There are no administrative facilities to attract experts to analyze tests' questions and to interpret results.

Recommendations in development of educational tests in programming which are given in this work can be used as based ones in different universities where disciplines which are connected with algorithmic languages and programming are taught.

REFERENCES

[1] H. Gulliksen, "Theory of mental tests", New York: John Willey & Sons, Inc., 1950.

[2] C. McKenna, J. Bull, "Designing effective objective test questions: an intoructory workshop", CAA Centre, June 1999

[3] V.V. Podbelskiy, O.V. Maksimenkova. "Programming as a part of the Software Engineering education" // Proceedings of the 4-th Spring/Summer Young Researchers' Colloquium on Software Engineering (SYRCoSE 2010), 2010. 165 – 168 pp.

[4] T. Dawson, "Basic concepts in classical test theory: relating variance partitioning in substantive analyses to the same process in measurement analyses", URL: http://www.eric.ed.gov/PDFS/ED406443.pdf

[5] В.И. Звонников, М.Б. Челышкова. Современные средства оценивания результатов обучения. – М.: Издательский центр «Академия», 2009. – 224 с.

[6] В.И. Звонников, М.Б. Челышкова. Контроль качества обучения при аттестации: компетентностный подход. – М.: Университетская книга; Логос, 2010. – 272 с.

[7] Sun Haiyang "An application of Classical test theory and Manyfacet Rasch measurement in analyzing the reability of an English test for non-English major graduates", Chinese Journal of applied linguistics (Bimonthly). China, vol. 33, pp. 87 – 102, April 2010.

[8] Т.П. Хлопова, Т.Л. Шапошникова, М.Л. Романова, А.Р. Ушаков. Математические модели дидактического процесса // Научно-теоретический журнал «Ученые записки». – 2010 – № 2. с. 107 – 112.

[9] J. Gonzalez, R. Wagenaar, "Universities' contribution to the Bologna Process. An introduction", Spain: Publicaciones de la Universidad de Deusto, 2008.

[10] URL: http://www.proftesting.com/test_topics/pdfs/steps_9.pdf

[11] S. Varma “Preliminary item statistics using point-biserial correlation and P-values”, URL: http://www.eddata.com/resources/publications/EDS_Point_Biserial.pdf

[12] P.V. Engelhardt “An Itroduction to Classical Test Theory as Applied to Conceptual Multiple-choice Tests”