REFERENCES

1. Ob obrazovanii v Rossiiskoi Federatsii [Elektronnyi resurs]: fe-der. zakon ot 29 dek. 2012 g. № 273-FZ. [On education in the Russian federation]. Moscow, 2012. 404 p. URL: http://minobrnauki.rf/documenty/2974/file/1543/12.12.29-FZ_Ob_obrazovanii_v_Rossiyskoy_Federatsii.pdf, (Accessed: 19.12.2016).
2. Gosudarstvennaya programma Rossiiskoi Federatsii «Razvitie obrazovaniya» na 2013–2020 gody [Elektronnyi resurs]: utv. rasporyazheniem Pravitel'stva Ros. Federatsii ot 15 maya 2013 [State Program of the Russian Federation 'Development of Education' approved by the Government of the Russian Federation]. №792-p. [Moscow, 2013].700 p. URL: http://minobrnauki.rf/dokumenty/3409/file/2228/13.05.15-Gosprogramma-Razvitiye_obrazovaniya_2013-2020.pdf, (Accessed: 19.12.2016).
3. Livanov D.V., Volkov A.Ye. Stavka na novoe soderzhanie [Elektronnyi resurs]. Vedomosti [Bulletin]. 2012. № 3179. URL: http://www.vedomosti.ru/opinion/articles/2012/09/03/stavka_na_novoe_soderzhanie, (Accessed: 19.12.2016).
4. Konanchuk D., Volkov A. Epokha «Grinfilda» v obrazovanii [Epoch of 'Greenfield' in education] [Elektronnyi resurs]: SEDeC; Center of education development of Moscow Management School SKOLKOVO (SEDeC). Skolkovo, 2013. 52 p. URL: http://www.skolkovo.ru/public/media/documents/research/education_10_10_13.pdf, (Accessed: 19.12.2016).
5. Mityushov E.A., Berestova S.A. Teoreticheskaya mekhanika: ucheb. [Theoretical mechanics]. Moscow: Akademia, 2006. 320 p.
6. Mityushov E.A., Berestova S.A. Teoreticheskaya mekhanika: ucheb [Theoretical mechanics]. 2-e izd. Moscow: Akademia, 2011. 320 p.
7. Belayeva Z.V., Berestova S.A., Denisov Yu.V. et al. Teoreticheskaya mekhanika v primerakh i zadachakh [Theoretical mechanics in examples and problems]. Mityushov E.A. (eds.). Moscow: Akademia, 2012. 176 p.
8. Berestova S.A. Proektirovanie obshcheinzhenernogo modulya programm proizvodstvenno-tekhnologicheskogo bakalavriata [Project of general-engineering module of production-technology Bachelor programs]. Inzhenernoe obrazovanie [Engineering education]. 2014. № 14. P. 100–105.
9. Rebrin O.I., Sholina I.I., Berestova S.A. Interdisciplinary project for Bachelor Engineering Program [Electronic resource]. Sharing successful engineering education experiences: 10th Int. CDIO Conf., Barcelona, Spain, June 16–19, 2014: Proc. Barselona, 2014. [9 p.]. URL: http://www.cdio.org/files/document/cdio2014/14/14_Paper.pdf, (Accessed: 12.12.2016).

UDC 378.146

Monitoring Math Competency of IT Students

Tver State University
S.M. Dudakov, I.V. Zakharova

The paper studies a method to develop testing and assessment materials, which is based on splitting "classical" parts of mathematics into smaller disciplines. It reveals the opportunities of the method in terms of competency-based approach.

Key words: competency approach, testing and assessment materials.

Introduction

Competency approach implemented into educational standards challenges many education programme designers. One of the challenges is competency mapping and developing relevant testing and assessment materials, which can be used at each stage. This paper describes how to overcome this challenge and simultaneously solve many other tasks and difficulties, which education programme designers normally face.

Mathematical disciplines for IT students

Training high-qualified IT professionals is impossible without profound mathematics education. Unfortunately, in the Russian language we still lack an appropriate term to call this section of mathematics. English "Computer Sciences" can be interpreted in different ways – either literally or transforming into "theory of IT", "IT fundamentals"; "math fundamentals for IT", etc. Nevertheless, one can identify the key subfields which are consistently incorporated into this mathematical course: different areas of discrete maths (theory of graphs, theory of Boolean functions, theory of coding, etc.), formal languages and automata theory, mathematical logic, algorithms, some topics of general algebra. Other disciplines incorporated into this course are optional and depend on a particular education programme, for instance, calculus of probabilities or numerical mathematics (see [1, 2]). Other mathematical subfields, if included, are regarded as supportive (for instance, calculus), since relevant knowledge and skills are also developed within other disciplines but rarely used in professional activities.

The incorporated disciplines are interrelated: either one discipline follows the other discipline or they are mutually influence each other. In this latter case, the contents of two disciplines are interdependent, which challenges programme and curriculum design. A typical example is mathematical logic and algorithm. On the one hand, fundamentals of both disciplines can be taught independently, but if to go deeper, one can see that these disciplines are strongly connected and can be taught as one subject. However, this method does not always work and a dozen of disciplines can never be taught as the only one. The probable decision is to teach several subjects simultaneously, but this is quite challengeable in practice. Another option is to split the discipline into smaller sections, which can be taught in different time. First, fundamentals, which are essential for other disciplines, are taught, and later it is possible to go deeper. However, the problem is that by the time, when the subject is learned in detail, students will have already forgotten some of the fundamentals, and the teacher and students have to spend extra time revising.

Another challenge is to motivate student to learn theoretical subjects. It is no secret that many students want to quickly acquire professional skills without learning fundamentals. It is particularly true for the first year students due to the decreased level of training at schools and no occupational

S.M. Dudakov

I.V. Zakharova

guidance for school children. And here one faces the same challenge again: on the one hand, it is logical to start with fundamentals and then pass to practical application, but on the other hand, IT students have low motivation for studying theory and sometimes possess insufficient background knowledge of math. To solve this problem, one can follow the ways mentioned above: unite several subjects within one discipline, study several subject simultaneously, or study the same subject several times (at different levels).

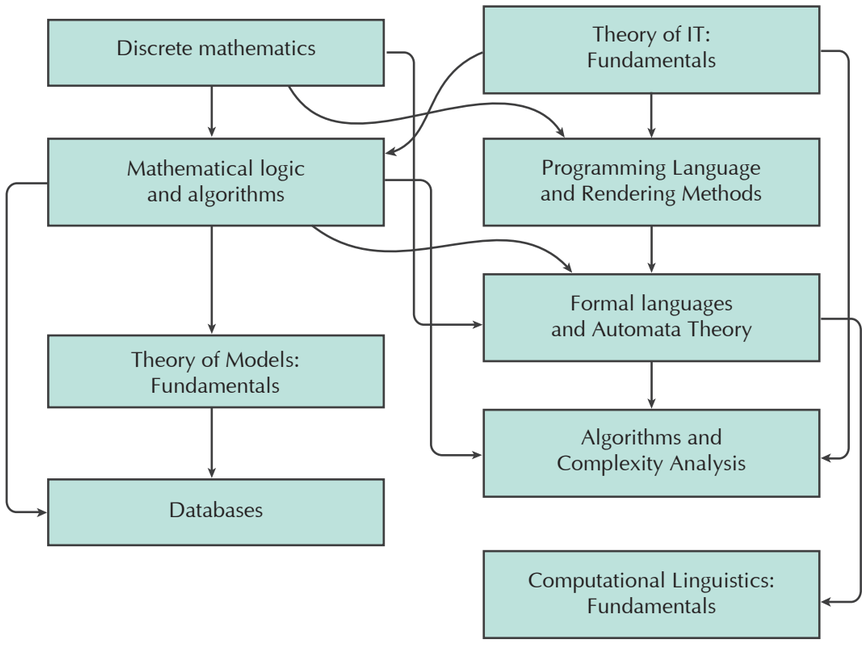
We suggest using the model shown in fig. 1 (see also [1]) but do not go as far as stating that it is ideal. As one can see, Discrete Mathematics and IT Fundamentals are introductory courses and provide only basic knowledge, which students can deepen over the following years of study.

Development of Assessment Materials

Testing and assessment materials for the model given above are similar for different subjects since the concepts are often the same and students have an opportunity to revise the material studied.

Moreover, this contributes to resolving one of the tasks which an education programme designer faces within competency approach (see [3]). Competency mapping is an essential stage in designing education programme within the competency approach. This map describes each competency elements and learning stages. Testing and assessment materials are designed to monitor competency development at every stage. The suggested model implies that within each section there are tasks at different levels of complexity (from elementary to more complicated),

Fig. 1. Dependence between IT fundamental disciplines



which can be represented as a graph of competency (or its element) development.

Let us consider several cases. One of Bachelor's competencies within professional profile 02.03.02 is PK–2: an ability to comprehend, develop, and apply up-to-date mathematical tools, concepts, and methodologies, international and professional IT standards. This competency is complicated enough and implies studying a lot of materials. Let us describe how to monitor the development of the ability to comprehend, develop, and apply up-to-date mathematical tools, one of the elements within the given competency.

The section devoted to formal grammar and automata is repeated within the model at least five times:

- learning fundamental concepts in "Discrete Mathematics";
- application of grammars and automata for description of a programming language and rendering in "Programming Language and Rendering Methods";
- advanced level course "Formal languages and Automata Theory";
- application of automata for text processing in "Algorithms and Complexity Analysis";
- application of context-sensitive grammar for automatic analysis and language processing in "Computational Linguistics: Fundamentals".

This example shows that courses providing theoretical knowledge on formal grammars and automata interchange with those implying practical application of the knowledge obtained.

The relevant testing and assessment materials allows monitoring the level of formal language competency development:

1. Basic level, Discrete mathematics – simple tasks on designing finite-state machines, regular expressions, and the simplest properties of regular languages:
   - To design a finite-state machine recognizing the definite language.
   - To design a regular expression describing the definite language.
   - To prove that the definite language is not regular.
2. Intermediate level, Programming Language and Rendering Methods - tasks on designing lexical and syntactical analyzers, the simplest context-free grammars and nested stack automata:
   - To design grammar for the definite programming language.
   - To design a lexical analyzer.
   - To design a syntactical analyzes.
3. Intermediate level, Formal languages and Automata Theory – tasks implying specific knowledge about languages and their properties:
   - To design grammar, equivalent to the given one, in strong Greibach normal form.
   - To prove that the given language is not context-free.
4. Advanced level, Algorithms and Complexity Analysis – tasks on designing efficient text processing algorithms:
   - To design a string-matching automaton based on Morris-Pratt algorithm.
   - To implement linear-time algorithm for a two-way nested stack automaton.
5. Advanced level, Computational Linguistics: Fundamentals – tasks on designing grammars more complicated than context-free ones:
   - To design dependency categorial grammar for the definite language.
   - For the definite language, to design the grammar which does not parse expressions.

Let us consider another example involving formal algorithm and calculation. Once again, it is the section that students face several times:

- Theory of IT: Fundamentals – the simplest models of programming languages (structured, functional, etc.) and their equivalence, some experience in computational complexity.
- Discrete Mathematics – Turing machines and model simplification, as well as undecidable problems.
- Mathematical logic and algorithms – review on mathematical modelling,

algorithms, and their properties, fundamentals of the theory of recursive and recursively enumerable sets, complexity classes.

- Algorithms and Complexity Analysis – different calculation models to design an efficient algorithm, complexity classes (studied earlier) for task analysis.

This sequence enables developing PK-2 competency in terms of another element – decidability and computational complexity:

1. Basic level, Theory of IT: Fundamentals – tasks on designing algorithms in different languages and their transformation:
   - To design a structured programme for the given task.
   - To transform a structured programme into a functional one.
2. Basic level, Discrete Mathematics – tasks on algorithm transformation and the simplest tasks involving undecidable problems:
   - To demonstrate that in any Turing machine it is possible to eliminate operations preventing the machine from moving its head.
   - To prove that the problem of reachable state for Turing machine is undecidable.
3. Intermediate level, Mathematical logic and algorithms – tasks on designing different types of algorithms, their transformation, undecidable problems, computational complexity:
   - To design a definite partially recursive function.
   - To design a counter machine for this function.
   - To prove that the definite set is m-full.
   - To design a cellular automaton recognizing the definite sign taking time linear in input size.
4. Advanced level, Algorithms and Complexity Analysis – tasks on computational complexity:
   - To prove that the problem is NP-complete.

One more example of information repeated is knowledge about logic programming languages:

- Discrete Mathematics – first insight into languages of propositional and predicate logic at the semantic level.
- Mathematical logic and algorithms – more difficult programming languages, formal inference, interconnection between logic and calculating.
- Theory of Models: Fundamentals – complex concepts linking syntax and semantics in programming languages.
- Databases – application of Programming Language Theory in query parsing.

Let us describe the stages in development of PK-2 competency in terms of programming languages:

1. Basic level, Discrete Mathematics – tasks on knowledge representation in programming languages, simplest transformations:
   - To determine the formula of predicate logic describing the definite properties of the entity.
   - For the definite propositional logic, to design the truth table, conjunctive and disjunctive normal forms, Zhegalkin polynominal.
2. Intermediate level, Mathematical logic and algorithms – tasks on inference and modelling:
   - To obtain the definite computation sequence.
   - To define the distributive law in Peano axioms.
   - To enrich algebraic system so that the definite formula will be true.
3. Advanced level, Theory of Models: Fundamentals – task on connection between syntax and semantics:
   - To prove that the definite property of the entity fails to be described by universal formulae.
   - To identify 1-types in the definite system.
4. Advanced level, Databases – tasks on composing queries by means of relational algebra and inexpressibleness:
   - To build a query in relational algebra with fixed-point operator, which returns all elements with the definite properties.
   - To prove that it is impossible to describe the definite property in relational algebra without a fixed-point operator.

PARTICULARITIES OF EDUCATIONAL TECHNOLOGIES

ENGINEERING EDUCATION
21'2017

Conclusion

We have considered opportunities which arise from splitting big educational sections into smaller parts studied at different time. This strategy allows education programme designers to overcome both typical challenges and those faced within competency approach.

REFERENCES

1. Zakharova I.V., Dudakov S.M., Yazenin A.V. O razrabotke primernogo uchebnogo plana po UGNS "Komp'yuternye i informatsionnye nauki" v sootvetstvii s professional'nymi standartami [On the development of curriculum on UGNS "Computer and Information Science" in accordance with professional standards]. Vestnik Tverskogo gosudarstvennogo universiteta. Seriya: Pedagogika i psikhologiya [Bulletin of Tver State University. Pedagogy and Psychology], 2016, no. 2, pp. 84–100. (In Russ., abstr. in Engl.).
2. Zakharova I.V., Dudakov S.M., Yazenin A.V. O razrabotke magisterskoi programmy po UGNS "Komp'yuternye i informatsionnye nauki" v sootvetstvii s professional'nymi standartami [On the development of Master's programme on UGNS "Computer and Information Science" in accordance with professional standards]. Vestnik Tverskogo gosudarstvennogo universiteta. Seriya: Pedagogika i psikhologiya [Bulletin of Tver State University. Pedagogy and Psychology], 2016, no. 3, pp. 114–126. (In Russ., abstr. in Engl.).
3. Zakharova I.V., Dudakov S.M., Yazenin A.V., Soldatenko I.S. O metodicheskikh aspektakh razrabotki primernykh obrazovatel'nykh programme vysshego [Developing exemplary education programmes for higher professional education: methodological aspects]. Obrazovatel'nye tekhnologii i obshchestvo [The Journal of Educational Technology & Society], 2015, vol. 18, no. 3, pp. 330–354. (In Russ.).