Journal of Physics: Conference Series

PAPER • OPEN ACCESS

Mastering software engineering with the help of the level model of competencies acquisition

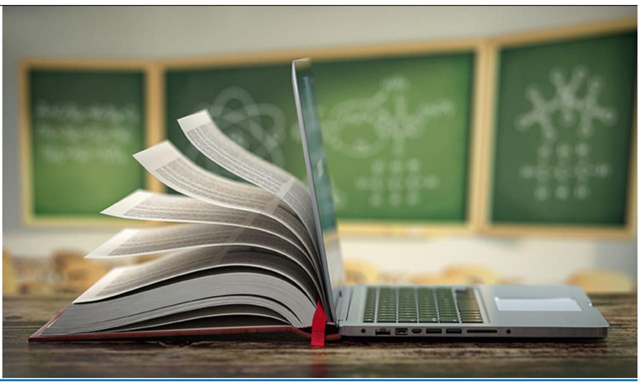
To cite this article: K A Rutskaya *et al* 2020 *J. Phys.: Conf. Ser.* **1515** 022087

View the article online for updates and enhancements.

The Electrochemical Society
Advancing solid state & electrochemical science & technology
2021 Virtual Education

Fundamentals of Electrochemistry: Basic Theory and Kinetic Methods
Instructed by: **Dr. James Noël**
Sun, Sept 19 & Mon, Sept 20 at 12h–15h ET

Register early and save!



This content was downloaded from IP address 2.50.129.40 on 28/07/2021 at 08:25

Mastering software engineering with the help of the level model of competencies acquisition

K A Rutskaya[1], O V Znamenskaya[1], V N Rutskiy[1], R Yu Tsarev[1,2], V N Everstova[3], I V Gogoleva[4] and A K Owais[5]

[1] Siberian Federal University, 79, Svobodny pr., Krasnoyarsk, 660041, Russia
[2] Krasnoyarsk State Agrarian University, 90, Mira ave., Krasnoyarsk, 660049, Russia
[3] North-Eastern Federal University, 48, Kulakovskogo street, Yakutsk, 677000, Russia
[4] Yakut State Agricultural Academy, 3 km. house 3, Sergelyakhskoye sh., Yakutsk, 677007, Russia
[5] Al Ain University, P.O. Box 112612, Abu Dhabi, UAE

E-mail: tsarev.sfu@mail.ru

Abstract. The paper proposes the use of an activity-based approach to acquiring the competencies of software engineering with the help of the author's model of level acquiring professional competencies. The approach allows the teacher to organize training software engineering so that students acquire needed knowledge, skills and abilities on the basis of a student's independent productive action aimed at mastering the levels of organization of thinking and professional activities of the software engineer in the process of designing and developing software products. The author's model is implemented to design level assignments for students to master some basic concepts of a software engineering course.

1. Introduction

The current situation in the global industry of professional software engineering and development is characterized by two mutually opposing trends:

- catastrophic ("hyper-exponential", according to experts of the Journal "Computerra-online") complication of information systems, communications and relations, due to the complexity of arranging distributed multi-user environments.
- reduction of most classical professional tasks used in mass training software engineering and development to the use of standard tools for developing information environments and application libraries.

This means that modern training software engineering and development requires:

- organization of teamwork, with the retention of the system hierarchy of the information system being engineered, developed or maintained;
- rapid acquiring of engineering and development tools and libraries, with an understanding of the high probability of their updating and ageing.

The problem of the "human-machine interface", which was formulated back in the 1960s, by expert and information systems engineer J. Weizenbaum [1] and science fiction writer S. Lem [2]. Traditionally, it is resolved by trying to adapt the interface to everyday models of human behaviour and perception. However, this leads to a kind of anthropomorphic perception of anthropomorphic systems and possibly to stress and even disasters when these systems behave not anthropomorphically,

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

but in their own logic predetermined by the program. Consequently, in the framework of mass education, it would be more productive to master patterns of thinking that would predict the behaviour of information systems and use them as efficiently as possible. Thus, there is an urgent need for the mass formation of information literacy among the students, a kind of “intuition of engineer and developer”, associated with understanding the internal logic of the work of programmable devices used both in solving professional problems and in everyday life.

In this regard, the fundamental importance of using tasks that require independent productive action in education is due to the fact that the student directly sees and can evaluate the result of his own action - instead of the teacher's external assessment in traditional types of tasks. We understand a productive action as “the ability and willingness to analyze problem situations, simulate activity patterns, build obviously non-existing or unknown activity patterns and act on these patterns” [3]. According to the level model of the formation of objective action developed in the framework of the cultural-historical conception of L.S. Vygotskiy, a productive objective action goes through three formation stages, and the transition between the stages is carried out through the development of a higher level of tools organizing activities [4].

The paper is aimed at particularization of the level model of mastering the objective action [4] in the format of the level model of software engineering competencies acquisition. Engineering as a subject material is valuable matter in that the very nature of the activity involves the creation of a final product that exists independently of the author: the program solves the problem with the help of a formal executor (a computer system, a robot, or at least a person acting strictly according to the rules).

The approach to constructing the contents of instruction in software engineering that is presented in the paper gives the student the opportunity to learn the specifics of software engineering as a type of professional activity from his own experience. The emphasis is not on particular technologies, models and engineering and development platforms, and especially not on the acquiring specific techniques for dealing with engineering and development environments or application software, but on the formation of key concepts and principles on which the engineer’s activities are based.

2. Approaches to the formation of competencies in software engineering

Let us turn to research on education in computer science. In a number of works, including works of L.L. Bosovoi, N.D. Ugrinovich, I.G. Semakin and others that underlie mass approaches to teaching the basics of programming and computer science in educational organizations, software engineering and development is understood as the art of controlling information processing machines.

At the same time, there is a broader approach. It is presented both in works devoted to the specifics of professional thinking of a programmer (as opposed to that of a mathematician and engineer) and professional training in programming (for example, authors such as E. Dijkstra, N. Wirth [5], A. D. Perlis, N. N. Neupeyvoda [6]), as well as in works devoted to the formation of a programming culture itself (S. Papert, A.P. Ershov and others) as an important part of modern culture as a whole. The great commonality of the approach is associated with the understanding that the processing of information and the choice of its presentation are instrumental in relation to solving a class of applied problems (tasks of scientific, engineering and economic computing, processing of texts and images, control of technical devices, and so on).

Therefore, the first step is to transform the class of applied problems (often solved at the level of intuition, unsystematized practical techniques) into formal models and the problems of converting these formal models represented by data structures and transformation rules. The next step is a description of the sequence of transformations based on the known rules and properties of data structures, the formation of an algorithm for solving the problem. And at the third step, the algorithm needs to be converted into a description of the sequence of actions, whether it is an instruction for a user of an information system, or a set of operations performed by an automatic device.

This implies the idea of three levels of thinking in software engineering and development:

- creation of an applied formal model and its description in the language of data structures, statement of the task (problem) in terms of a formal model (modelling);
- a description of the sequence of transformations of the formal model that together solve the problem (algorithmization);
- writing a direct sequence of instructions for the executor (coding; this level is traditionally considered programming in the public mind).

Note that traditionally, the subject of computer science appears for students as a solution to exercises on the algorithmization of traditional problems of arithmetic calculations and mastering specific methods of handling engineering and development environments or application software. E.A. Nigmatulina believes that “traditionally, programming training is carried out on training examples and tasks, linearly – from simple to complex” [7]. The difficulty level of the task is set either by the number of operations that must be performed in a standard situation or by the complex structure of the condition which makes the application of one or another solution algorithm non-obvious. Only tasks of the contest type (such as competition, olympiad or hackathon) require the study of the object and the application of rules that go beyond the standard educational program.

Such training methods imply the reproduction and transfer of known methods of action with predetermined material, the requirements for the assimilation of significant amounts of facts and information, for example, about the interface of applied engineering and development tools and environments of engineering and development. Namely, this information becomes outdated extremely quickly, in contrast to theoretical knowledge and general patterns of activity organization and is likely to become useless by the end of the learning process. The solution of problems, in this case, consists in applying the rules and definitions what enables the student to form such qualities as diligence, attentiveness, ability to makevolitional efforts – these things were traditionally defined as the main personal results of education according to classical scholars in the field of education, from J. Locke to I.-F. Herbart.

At the same time, in the modern world, according to the estimates of international and domestic expert centres (such as Future work skills 2020 [8], international competencies contests “WorldSkills”, including children's contests “JuniorSkills” and “FutureSkills”, competitions organized in Russia with the assistance of the Agency for Strategic Initiatives), other universal qualities come to the fore. Particularly we mean the ability to comprehend, to highlight the main thing, to set the goals, social intelligence, innovative and analytical thinking, interdisciplinary literacy [9].

The development of such competencies is based on two key parameters: the transition from mastering factual information to the mastering knowledge of technology, the transition from the ability to perform standard tasks in the framework of established rules to the ability to act in a non-standard situation, synthesizing new knowledge. When correlating this with the description of the structure of activities of the software engineer, it can be argued that the competency-based approach related to the inclusion of a person in productive activity, or at least inclusion of a person in imitation of productive activity in the form of completing educational tasks, is very adequate for such disciplines as computer science (as well as for disciplines, related to applied engineering).

For the formation of students' intellectual skills, we need an active approach to learning aimed at the formation of key software engineering and development concepts, the development of thinking in the process of mastering new knowledge, and the formation of productive actions to transform the subject material studied. The result of such training is a holistic vision of the subject, not as a set of knowledge about it, but as a system of principles and means that regulate, in particular, the construction of fundamentally new content (scientific discoveries, engineering developments etc.). In the framework of this approach, students get the opportunity to learn the specifics of programming as an activity from their own experience.

3. A model for the level acquiring competencies in software engineering

We have chosen a level model for designing training that ensures individual student progress in mastering the subject content as the basisfor designing the model of acquiring competencies of software engineering. In this case, individual progress is perceived as reaching a higher level of mastering subjective actions, which is determined by the type of their mediation (B. D. Elkonin, B. I. Khasan, A. M. Aronov, O. V. Znamenskaya, etc.) [4, 10, 11].

According to this model, any subjective action in its formation goes through three levels of acquiring. The first level is the mastery of the general meaning and form of action. At this level, the action is mediated by a rule, a template, a known algorithm and is correctly performed only in a standard situation. At the second level, the approximate basis of the action is the essential attitude, the general principle of the action. Orientation to general principles and methods of action allows it to be implemented in a so-called «noisy» situation when the application of rules and algorithms is not obvious. The transition to the third, highest level of mastering the action is characterized by the inclusion of a generalized mode of action in the student’s personal resources. The action becomes productive essentially such that the generalized method, borrowed from a known subject, is appropriate and acts as a transformative material for another subject. In new situations different from the formation situation this allows to accept and reject, correct and transform the very essential basis of the mode of action, what requires high-level mental reflection, analysis, synthesis and generalization [4].

The Delta toolkit was developed based on the level model in order to diagnose the level of individual progress in studying courses “Russian language” and “Mathematics” in educational organizations (O.V.Znamenskaya, O.I.Sviridova, L.A.Ryabinina and others) [4, 10, 12] and in the course of SAM testing [11]. Based on the model, teacher training programs are implemented for theacquiring of these tools and the design of learning conditions conducive to individual student progress and achieving personal, subject and meta-subject results (K.A. Bazhenova, O.I.Dyatlova, O.V. Znamenskaya, O. I. Sviridova, L. A. Ryabinin and others) [4, 10].

Samples of educational and diagnostic materials are developed based on the level model for the training courses “Mathematics”, “Russian Language”, “Physics”, “Biology”. The authors of these training materials on the course of mathematics (A. M. Aronov, O. V. Znamenskaya, K. A. Bazhenova, V. G. Likontseva, O. A. Franzen, etc.) [4, 12] distinguish an algorithmic line in the course mathematics, however, it does not cover the system of key concepts and substantive actions of informatics and programming.

We assume that the use of a level model as a means of analyzing the structure of programming activities allows us to develop a competency-based model for teaching software engineeringas part of a computer science course.

We specify the levels of mastering the subjective action, taking into account the peculiarities of creating software as a type of professional activity of a software engineer.

First level. Reproduction of the sequence of actions according to the model or the established scheme, ideally, brought to automatic action. The student can perform specific operations (possibly quite complicated) to solve the standard task according to a ready-made algorithm using standard procedures. He can identify the conditions partially in which the execution of a sequence of actions is difficult.

When mastering the activity of programming, tasks of the first level are associated with the development of basic coding and algorithmization procedures, the rules for using at least one programming language: before undertaking complex tasks, a student must master at least one tool to solve these problems. Training tasks at this level can be aimed at compiling an algorithm for another human performer and understanding the differences between a human performer and a machine.

Second level. The solution of a class of tasks oriented to constructing an action on the basis of a combination of patterns of solving the problem, including the analysis of conditions and the selection of necessary tools. The student can formulate the task as a result of analyzing the problem situation, choose a method that meets the conditions and limitations of the task, and also evaluate the adequacy of the result of the action.

Learning tasks at this level require the ability to design and combine algorithms, translate them into code, justify the efficiency of the algorithm and its effectiveness for solving the task. Actions at this level imply identifying the general principles of working with the concept of “program” and detecting significant differences between the concepts of “algorithm” and “program”. At this stage, the student knows the semantics of programming languages, can divide the task into subtasks, build a complex program as a set of simpler ones, use a library of algorithms.

N.N. Nepeyvoda notes that “the student’s practical goal is to learn to see general constructions in the language even without preliminary learning it” and “understand the programs written in it” [6]. Thus achieving the second level of acquiring of subjective actions is supposed to be a subject result of training.

The third level of mastering programming activity implies the construction of a method for solving a problem in conditions where the method is absent or unknown. The student is able to combine and transform the methods learned for solving the problem using theoretical knowledge and objective experience, turn the problem into a task, formulate sub-tasks based on the analysis of the problem situation, and build a sequence of actions for solving them. He or she is capable to determine the level of difficulty of the task, the similarities and differences with the known types of tasks.

Training tasks at this level may require the use of well-known algorithms and programming languagesto solve applied modelling problems in a specific subject area. A feature of the task material for mastering the third level of subjective action is the need to operate with information from the subject area in which the task was originally posed. Thus, a productive action requires the retention of inter-subject communications necessary to solve the problem.

4. An example of the design of level assignments for mastering some concepts of a software engineering course

As for an example of an assignments system developed on the basis of the author’s level model of competencies acquisition, we present a series of assignments (tasks) aimed at mastering the concepts of “algorithm” and “program” as part of a programming course created by the authors together with K.V. Morozova [13]. The system of tasks is arranged similar to the tasks of the algorithmic diagnostics "Delta", designed to master the course of mathematics [12]. The series can be used by the teacher both for diagnosing the level of mastering the subject action by students, and for the formation of these concepts also when developing adaptive electronic educational courses for teaching software engineering and development [14].

A series of tasks contains preliminary text, 2 tasks at the first level, 2 tasks at the second level and 3 tasks at the third level of competencies acquisition, as well as two two-level tasks which level is determined by the method of solution demonstrated by the students. The preceding text describes the plot of a series of tasks, provides the necessary algorithms for completing tasks. Fulfillment of tasks requires students to retain the basic characteristics of the algorithm (effectiveness, discreteness, determinism), and to highlight the essential condition for the transition from the algorithm to the program – the presence of a formal executor. This is due to the fact that programming activity was initially associated with solving applied tasks, gradually alienating the algorithm of actions from a human executor and understanding the essential conditions for implementing a program, thus gradually transferring instructions from an informal executor (human) to a formal one (machine).

Tasks at the first level (1.1. and 1.2. In [13]) reveal whether the student understands the boundaries between the formal (computer) and the informal performer (human). The first level corresponds to such a decision of a student when he or she can reproduce the algorithm in accordance with the pattern. The mediator in solving problems is a pattern, rule, template or algorithm. When completing the task, it is not necessary to go beyond the scope of the pattern, it is enough to follow the indicated actions formally. The task is considered completed if the original algorithm is performed by students correctly.

Task 1.1. Draw a letter according to the algorithm written for the draftsman (the first quarter of the coordinate plane on checkered paper is proposed): Move to the point (3; 1); Lower the pen; Move to the point (3; 4); Move to the point (5; 4); Move to the point (5; 1).

Task 1.2. Write the program "Letter B", which is depicted on the coordinate plane (a place for answers and a picture depicting the algorithm for moving the pen of a draftsman are offered).

Two-level tasks (tasks 1.2.1., 1.2.2. in [13]) can be performed both at the first and at the second level of mastering subjective action [4]. A multi-level task is considered to be completed at the first level if the student uses the algorithm as an informal performer, not taking into account the fact that the algorithm needs to be compiled for the performer-draftsman. The task is completed at the second level if the student relies on an analysis of the performance of actions by the formal executor when substantiating his choice.

Task 1.2.1. Draw the elements of the letter "Yu". What other letters can be drawn from these elements? Write an algorithm for a draftsman, which includes images of one of the elements of the letter "Yu". Come up with several options. (the checkered field is suggested).

Task 1.2.2. An algorithm is proposed for a human performer. Write an algorithm for a draftsman (an algorithm consists of 10 elementary actions that a draftsman can perform [13]). Are the algorithms different for a person and for a performer? How?

The solution of tasks at the second level (see tasks 2.1. in [13]) implies understanding of the differences between the algorithm and the program. Tasks at the second level allow you to determine whether the student is able to detect the sequence of steps in the algorithm, to operate on the information about the set of commands that the executor can execute, to present an image of the future result after the algorithm is completed, to highlight the general way of writing letters by the performer-draftsman. The task is considered to be completed at the second level if the algorithm is correctly and fully completed according to the given instruction in new conditions that go beyond the applicability of the instruction, and a significant difference in conditions is found that does not allow a student to apply the instruction learned directly.

Task 2.1. Find out which letter you can draw using the algorithm (the algorithm is given [13], which represents a sequence of elementary actions, two of which are omitted. The student must reconstruct them himself or herself). *Recover the missing actions. Can a draftsman solve this problem? Will he or she get the same result as you? Justify your choice.*

The solution of tasks at the third level (for example, task 3.1., 3.2. in [13]) implies the use of coding to engineer and develop a program that meets certain conditions (for example, the algorithm should be optimal in the number of operations). The assignment requires applying the general principle of constructing a letter from its components to creating part of the code in the algorithm for individual elements and retaining the alphabet as a system of elements. A student performing tasks at the third level is able not only to see the boundaries of the application of the new algorithm defined earlier by the pattern but also to transform this algorithm in accordance with changes in the material.

Task 3.1. You need to teach the draftsman to draw the letters "R", "A", "F". Write the general part of the algorithm for the draftsman so that it can be used for all these letters. Add additional steps to the ready-made algorithm to draw each letter individually.

Task 3.2. Write a program that would draw letters using a milling machine. Make a base of basic elements for drawing letters.

5. Conclusion

The authors' approach to the analysis of the structures of software engineering activity allows us to identify some key concepts (algorithm and program) that are necessary for studying software engineering and computer science. The use of the level model [4, 13] gives the basis to design a series of level assignments for acquiring these concepts by students. The system of assignment types specified in the paper can be used both in constructing the training content in an active approach and as a material for identifying the type and intensity of individual progress of students in mastering software engineering and other computer science courses.

An active approach to the formation of the concept of "programming" on the basis of a level model of competencies acquisition forms the basis for constructing meta-subject concepts, and, therefore, for developing key human abilities to analyze an atypical situation and engineer a way of acting in it. The focus on the formation of key concepts and principles of software engineering when formulating the tasks of training teachers in software engineering and computer science [15] allows us to expand the scope of the methodology of relevant teaching and requires an active approach to understanding the problem.

References

[1] Weizenbaum J 1982 *Possibilities of computers and the human mind. From judgment to computation* (Moscow, USSR: Radio and communication)
[2] Lem S 2002 *Sum of technology* (Moscow-St. Petersburg, Russia: AST-Terra Fantastica)
[3] Ermakov S V and Popov A A 2020 Additional mathematical education as a condition for the development of mathematical giftedness Retrieved from http://opencu.ru/page/dopolnitelnoe-matematicheskoe-obrazovanie-kak-uslovie-razvitija-matematicheskoj-odarjonnosti, last accessed 2020/02/10
[4] Znamenskaya O V, Sviridova O I and Ryabinina L A 2014 *Assessment-support of individual progress of students* (Krasnoyarsk, Russia: Siberian Federal University)
[5] Wirth N 1985 *Algorithms and Data Structures* (Upper Saddle River, New Jersey, US: Prentice Hall) p 288
[6] Nepeyvoda N N and Scopin I N 2003 *Foundations of programming* (Moscow-Izhevsk, Russia: Institute for Computer Research)
[7] Nigmatulina E A and Stepanova T A 2011 Conditions for the formation of an algorithmic culture of students based on the information approach *Bulletin of the Krasnoyarsk State Pedagogical University named after V.P. Astafiev* **1** 82-6
[8] Future work skills 2020. Report SR-1382A. Institute for the Future; the University of Phoenix Research Institute Retrieved from http://www.iftf.org/our-work/global-landscape/work/future-work-skills-2020, last accessed 2019/11/30.
[9] Popov A A, Ermakov S V and Remorenko I M 2015 Project "Assessment of competency-based results a achievements" *Collection "Open education as a practice of self-determination"* (Moscow, Russia: Non-profitpartnership "Author'sClub")
[10] Znamenskaya O V, Bazhenova K A and Skripka A M 2012 Towards the construction of a scheme for analyzing the educational conditions of individual progress of students *Proc. Int. Research and Applied Conf. "Theoretical and applied problems of science and education in the 21st century"* (Tambov, Russia: Business-Science-Society) pp 54-7
[11] Nezhnov P G, Kardanova E Yu and Ryabinina L A 2013 Study of the process of educational content acquiring *Issues of education* **4** 168-87
[12] *Monitoring individual student learning progress* 2006 (Krasnoyarsk, Russia: KPD Printing Center)
[13] Morozova K V 2016 *Series of level tasks in programming for students based on the model of individual progress* (Krasnoyarsk, Russia: Siberian Federal University)
[14] Tsarev R Y, Yamskikh T N, Evdokimov I V, Prokopenko A V, Rutskaya K A, Everstova V N and Zhigalov K Y 2019 An approach to developing adaptive electronic educational course *Advances in Intelligent Systems and Computing* **986** 332-41
[15] Aronov A M and Bazhenova K A 2015 Modern technology of formation of an active attitude towards the educational process among bachelor students in Pedagogics *Bulletin of the Southern Federal University. Pedagogical sciences* **11** 37-44