Learning advanced software development methods: problems and solutions

V.V.Kuliamin, V.A.Omeltchenko, O.L.Petrenko
{kuliamin, vitaliy, olga}@ispras.ru

Introduction

In the contemporary world the use of software in the economy, social and private life grows more and more. This fact leads to the constant growth in software functionality, complexity, and requirements to its safety and other quality characteristics. The problems of creation of complex and at the same time reliable software cannot be solved without advanced development methods based on the latest achievements in computer science.

However, the transition of these methods into industrial practice encounters a lot of problems, which can be classified in the following way.

1. Research problems related with the limitations of the method. The method can be applicable only to the software of certain problem domains or constructed on the base of certain architecture. In such cases the method needs significant modification to extend its applicability domain. Usually it requires research activities.
2. Technical problems related with integration between the method and the development tools used in the organization where the method is introduced. These problems can be solved by development of special tools or add-ons to some already available tools that allow the organization staff to use the method under transition in their traditional environment.
3. Organizational problems related with combination of traditional software development processes and management practices with new non-traditional method. Often, when traditional management, work distribution and performance metrics are used, an application of an advanced method decreases work efficiency or makes the picture of the development so paradoxical that the most managers with traditional skills can not work proper.
4. Cultural problems related with higher or simply nontraditional requirements to the staff posed by the advanced methods. The development becomes inefficient because of both failure of staff to meet new challenges and failure of management to find staff with required unusual skills.

The paper suggests an approach to solution of problems of the two last kinds, which are the most critical for technology transition in the industry. Technical and research aspects of advanced software development methods are considered only in the context of an adoption of new methods into industrial development.

If we want to solve the organizational and cultural problems, an essential drawback of advanced methods is too big cost of their acquirement caused by natural complexity of the problems of complicated multifunctional software development and lack of orientation to practical applications inherent to advanced methods. Usually these methods are the results of research activities; their initial representation is too abstract for most software developers and weakly connected to their vital problems.

In many cases complexity and acquirement difficulty of advanced methods are reasons that prevent their transition to industrial processes. Hence, successful practical using of them supposes preliminary *training* of users and mangers in effective use of these methods and tools supporting them. Managers should be trained also in effective approaches to organization of work according to new reality when using such methods, and adequate techniques of measuring work progress and other method-specific characteristics.

Training of different kinds is almost the only effective way of solution the organizational and cultural problems of advanced method transition. Among different kinds of training the following are definitely required.

1. University students should be taught as future staff of industrial software development

2. End users of tools supporting advanced software development methods should be trained to use them effectively
3. Managers should be trained in effective ways of project management in new environment
4. Persons responsible for development process refinement should be trained in ways of effective advanced method adoption
5. Broad promotion and popularization of advanced methods should be provided, including conduction of workshops and seminars for managers and technical staff

Model based testing methods are example of advanced software development methods. They provide means of much more effective control of quality of complex industrial software than traditional methods. In spite of this important advantage their transition in the industry is not easy. When adopting model based testing methods into industrial software development processes the same problems arise as when the most advanced software development methods are introduced in the traditional development.

This paper presents an approach of training advanced methods and a case study of its application to training in model based testing technology UniTesK. The UniTesK technology was developed in ISP RAS by the group of verification, specification and testing (RedVerst, [1]).

The paper is based on almost ten years experience obtained RedVerst group in using different learning methods when training testers, architects, managers, and university students in advanced model based methods of test development.

Learning goals and learning structure

The final goal of user training in new technology is the transition of this technology in industrial software development. To achieve this goal we should take into account different concerns and expectations and differences of comprehension of different groups of stakeholders:

1. *Students*, learning basic theoretical knowledge, which, as they hope, will be useful in their future work.
   For students very important thing is that their theoretical knowledge is being approved by practical experience in real projects, which can be provided by ISP RAS involved in academic research and industrial development. Having skills of applying knowledge on practice students could get more challenging work in their future.
2. *Software testers*, which should use acquired skills in real projects directly after training.
3. *Software architects and designers*, which should know, how the new technology changes their work, what should be done in addition to usual things, and how it changes the properties of the resulting product.
4. *Mangers*, which need knowledge about an economic effect of the application of new technology in different kinds of projects, and understanding, how the development process should be changed when applying new technology, what documents and activities should be added into the process, and what can be taken away, what metrics should be used to estimate the progress during the development and the result product.

Because in practice the only learning course can not serve all stakeholders several courses specialized to smaller groups of stakeholders have been developed:

1. Theoretical course on formal methods of programming and their use in testing with practical studies for university students.
2. Basic trainings of software testers in tools supporting model based method of test development — one training for one tool.
3. Advanced trainings of software architects and designers in tools supporting model based method of testing development.
4. Training of managers in project control when applying model based method of testing development.

Also distance learning is possible, when somebody investigates papers, documentation and other materials, but in this case it is important to have an active communication with learner and to

control learner's acquirement of required knowledge and skills. The Figure 1 presents the general structure of learning.

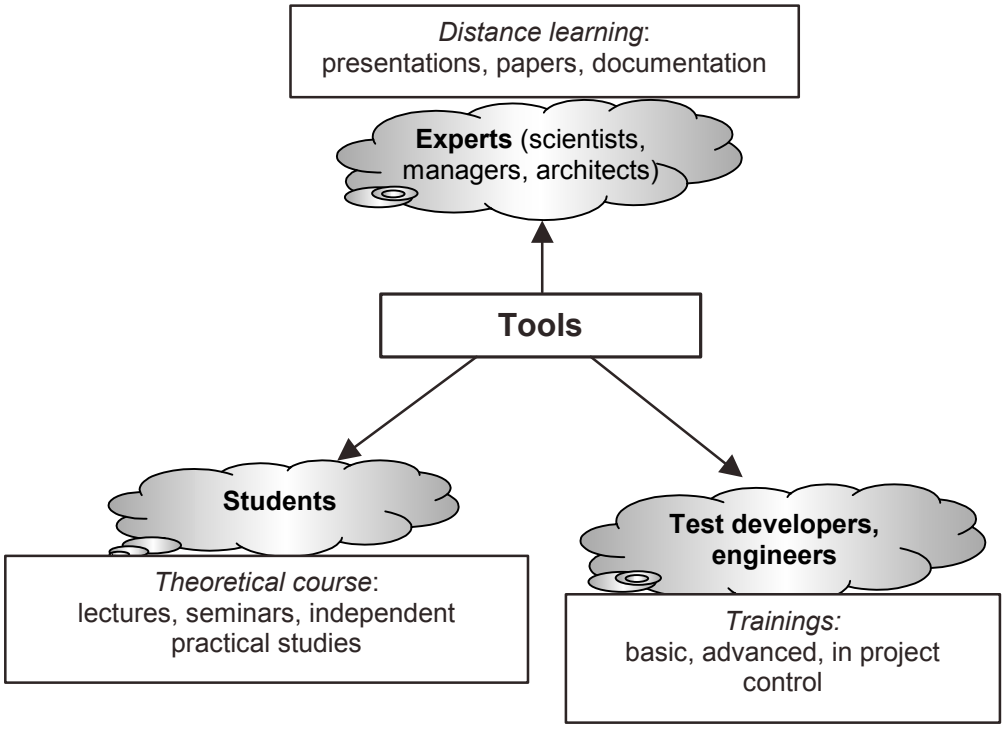

Fig. 1. General structure of learning.

Learning arrangement

Theoretical course

In syllabus of the Computational Mathematics and Cybernetics Faculty of Lomonosov Moscow State University there is the course "Formal specifications of programs", that is intended for 4th year students. The most impotent goal of the course is a demonstration of the last achievements in using formal methods in real software development. To meet this goal, on the one hand, the specification language should be used that have proved its practical usefulness in real projects. On the other hand, it should be shown how formal methods could be applied during different phases of software product life cycle — from software design to software delivering.
The first part of the course contains theoretical lectures providing to students knowledge about formal methods during studying RSL language. The following subjects are concerned:

- Formal specifications in an industrial software development
- Kinds of formal specifications
- RAISE Development Method and specification language RSL (RAISE Specification Language) [2].

The second part is dedicated to model based testing. It considers both general approaches of model based testing and a case study — UniTesK test development method that was developed in Institute for System Programming of RAS during 1994-2000 years. The theory of this course part is presented in three lectures: "Mathematic software models in testing", "Specification and oracles", "Problems of test sequence building".
UniTesK is the method of testing based on formal specifications. All tools of UniTesK family use common methods of software functionality description and unified test architecture. UniTesK tools differ in their target implementation languages — C, C++, Java or C#. The Figure 2 represents the main steps of test development in UniTesK method, for more details of UniTesK method see [3,4].

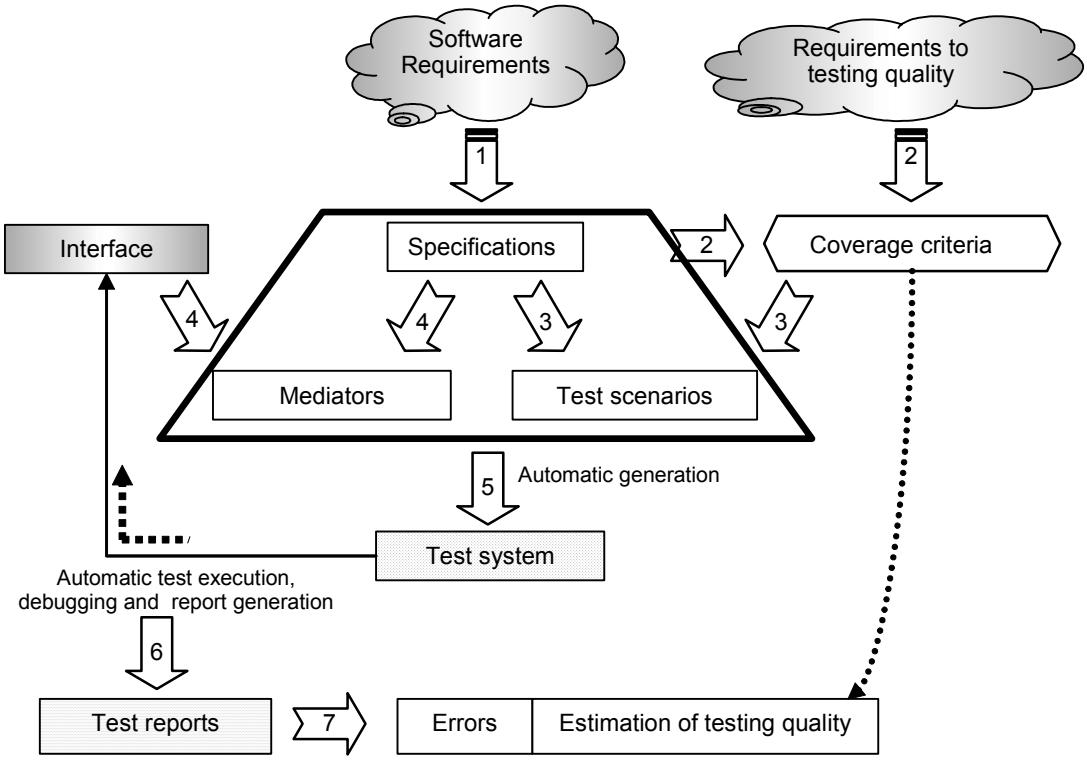

Fig. 2. UniTesK test development process.

The main feature of the course is a large practical part. During it students are studying at first the ways of specification development using RSL language, and then the testing tools that are used in real software development, for example J@T.

Students gain practical experience in ISP RAS. They are incorporated into the groups working on real projects and take a participation in the project part. Because usually students in spite of good enough theoretical basis have difficulties when applying their knowledge on practice each student has the project mentor managing student's work and ensuring real valuable results. In their diploma works students use the results produced during their work in real projects.

At the end of the course students have the following knowledge:

- how the mathematic models can be used in software testing,
- how test systems are design in cases of real complicated software,
- how testing are used in an industrial software development,
- how testing different kinds of software can be automated;

and skills:

- development of formal specifications of software functionality,
- development of mediators binding specifications and an implementation of the system under test,
- development of test scenarios,
- test execution and analysis of their results.

Specialized (optional) workshop "Program verification and validation" is dedicated to more detailed learning formal methods.

In Moscow Institute of Physics and Technology the review course «Formal specification languages» intended for 5th year students is conducted. It contains the following lectures:

- Formal specifications in an industrial software development
- Kinds of formal specifications
- Abstract data types and their using in specifications
- Invariants, axioms pre- and postconditions — the basis of specifications

- Modeling on formal specification languages, checking model's conformance
- Using formal specifications in software testing

At the end of this course students have knowledge about differences between formal specifications and programming languages, are able to differentiate structural and functional specification languages, understand concepts of modeling with use of formal specifications and their using in software testing.
Because this course is planned as the review course it has no practical part.
All mentioned courses have been developed by the staff of ISP RAS that have unique many years experience of using formal specifications both for description of functionality of industrial complicated software and analysis of software properties, and for test development based on formal specifications. Some persons of this staff are the leading experts in model based methods of test development in the world.
The second part of the course "Formal specifications of programs" is available under academic license for Europe, Asia and Australia.
Considered approach of student learning allows:

- achieving a well theoretical and practical level of proposed courses;
- fast acquainting students with the last advances in the given domain;
- acquiring practical skills useful in real work;
- during early phases of learning taking students capable for science work.

Training course

Learning industrial test developers in the same fashion as students is not possible, because in this case the major requirements are hard time constraints (a few days) and guarantee of results of learning. *Training course* has been used in this case.
Training course is a base course after that one can develop tests of programs. In the same time this course is a basis enabling later on to perfect skills of testing and to use competently co-verification processes in software development. Co-verification process [5] is a process of development of means of software correctness verification concurrently with the software development itself. Using such processes is a rather promising approach to the development of high quality software.
The goal of training is a production of skills required to specification based test development using UniTesK method.
To make possible rapid and high-quality acquirement of skills of model based test development training course is based on the following concepts:

1. Careful selection of training content in the correspondence of goals of the training;
2. Selection of adequate ways of learning;
3. Active and cooperative learning;
4. Creation of favorable learning environment [6].

Each concept has its implementation in considered training course. It allows to train during only 4 – 5 days with assured results.

Training content selection

The generalization method of training materials is used for content selection, during which one or several basic concepts are distinguished. They are later comprehensively and repeatedly developed during course using various approaches. In our case such concepts are the following main features of UniTesK:

- testing is considered as a comparison of properties of a software implementation behavior to model properties defined by formal specifications;
- system under test is considered as "black box", i.e. it is tested as a set of interfaces (operations, procedures, methods, data structures and so on), their internals — an implementation of them, algorithms, are not considered;

- specification are defined in the form of pre- and postconditions of operations and data invariants defining data integrity constraints;
- interfaces of an implementation and model could differ both in structure and in the abstraction level; binding the correspondent interfaces is provided by special program components called mediators;
- UniTesK offers the unified architecture of the test system, which implies the components of test actions' generation and behavior analysis of system under test as well;
- subsystem of test actions' generation is decomposed into two components: the generation of inputs of separate operations (parameter value iterators) and the generation of sequence of operation calls (test sequence);
- test sequence generation is based on the traversal of FSM constructed from operation specifications of and test scenario.

The training contents selected in compliance with these concepts is represented in a form of a training program. Figure 3 demonstrates an example of a program of the developed training in CTesK testing tool supporting UniTesK for C-language software.

Course contents

1. General target setting of conformance testing
Goals of testing, difference with "white testing". Main problems of test creation: oracle, test coverage criteria, regression testing.

2. CTesK technology overview
Specification-based testing. Test set architecture. Pre-build components. Generated components. Manually developed components. Technology steps: specification development, creation of mediators, scenario development, debugging, regression testing.

3. Development of specification of target software system
Definition of specification development. Implicit specifications. System model definition and formal definition of requirements for its functionality. Levels of abstraction and detailing.
System interface definition. Stimuli and reactions sets definition.
Data specifications: state of a system and data visible from outside. Definition of state. Types of parameters of stimuli and reactions. Invariants.
Stimuli specification. Specification structure. Pre- and postconditions. Postcondition verdict. Pre- and Post-states. Postcondition control graph view. Branch. Using variables before and after branch.
Types of coverage. Branch, coverage.

4. Creation of mediators.
Connection between model and implementation. Rise and fall of data abstraction levels (parameters and results). "Open" and "hidden state".

5. Scenario development
General scenario ideology based on FSM traverse graph. Final State Machine (FSM) as basic model of test creation. Definition of state and transition of machine. Factorization and problems concerned with it. Factorization development recommendations. General characteristics. Connection with test set architecture.
Set of procedures as set of FSM transitions. State. Iterators of parameters. Filters. Scenario options.

6. Debugging
Test system debugging steps. Optional tracings. Connection between test system components and specification.

7. Regression testing
Test run automation. Generation of reports. Modifications in test system with modifications in implementation. Comparison of test run results on different versions of implementation.

8. Test development tools
Test development in MS Visual Studio.

Fig. 3. Program of training in CTesK.

The acquisition of the knowledge mentioned in this program provides the necessary foundation for mastering of the following skills:

1. Testing of functions, the behavior of which does not depend upon the history of interactions between the system under test and its environment:
   - definition of function precondition;
   - definition of functional branches;
   - definition of function postcondition;
   - definition of type invariants;
   - definition of parameter iteration.
2. Testing of functions, the behavior of which depends on the history of interactions between the system under test and the environment, when the factorization is not required:
   - definition of function precondition;
   - definition of functional branches;
   - definition of function postcondition;
   - definition of type invariants;
   - definition of variable invariants;
   - definition of test state;
   - definition of parameter iteration.
3. Testing of functions, the behavior of which depends on the history of interactions between the system under test and the environment, when the factorization is required:
   - definition of function precondition;
   - definition of functional branches;
   - definition of function postcondition;
   - definition of type invariants;
   - definition of variable invariants;
   - definition of generalized state;
   - definition of parameter iteration;
   - definition of mapping generalized parameters to functions' ones.

Adequate learning ways' selection

Training course uses the method of "immersion" as the main form of learning. It means that during the training course (for 4 – 5 days) trainees are engaging only in training and are not looking aside for anything else.

In the training the following ways are used:

1. **Traditional straight lecturing** implies that a lecturer works with the whole class using the same pace and general tasks. This way is used to present new theoretical material required for instructional lectures, which directly precede the practical studying.
2. **Instructional lecturing** presents the technology of further activities, particularities of carrying out of certain actions, techniques, and algorithms of problem solving.
   Due to complexity of the course instructional lectures contain main concepts of the selected part of problem domain, their structure and interrelations.
3. **Practical studying** is carried out after significant parts of training have been studied. It is an independent trainee works with the possibility of intensive consultations with trainers.
   In the considered training the practical studies include specification development, test scenario development, test execution and debugging.
   During practical studying students' activity is managed by means of an independent choice of a sequence of steps of exercise performing and a self-rating of produced results. The sequence of steps could be composed from the intended set of ones:
   - Reading exercise
   - Independent exercise performing
   - Comparing produced results with the control answer

- Studying an algorithm of exercise performing
- Studying carrying out of a single algorithm step in performing of the given exercise

Figure 4 presents various possible sequences of these steps.

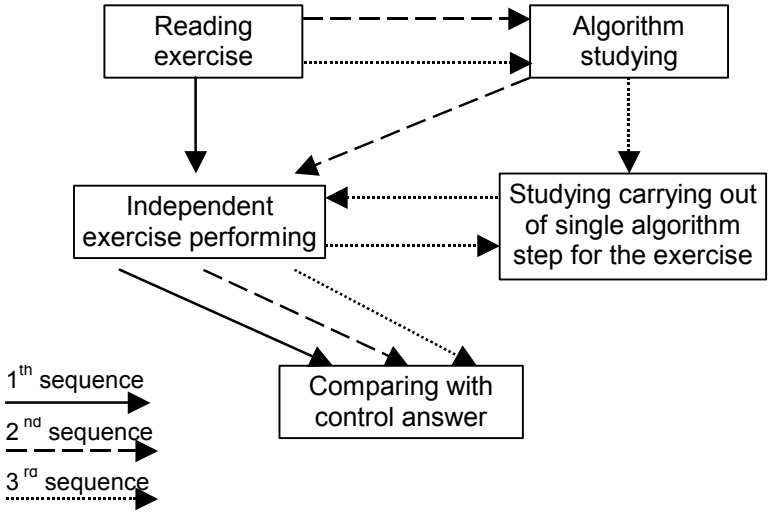

Fig. 3. Possible sequences of exercises' carrying out.

4. **Consultations of experts** (individual kind of training) are used as an accompanying training method aimed at clarification of certain problems as well as at more profound mastering of the subject.

Active and cooperative learning

In training industrial software developers it should be taken into account that trainees participate in various projects requiring creative thinking and skills of communication and cooperative activities. Therefore in training familiar and natural environment for their work is built.

In **active and cooperative learning** trainees are involved in active cognitive work. They discover the knowledge, which under traditional approach is presented by the teacher without active participation of the students themselves [7]. It is well known that when active and cooperative learning is used 75% to 90% of the information is acquired. (Compare: Traditional lecture yields acquirement of about 20% of the information introduced in it [8]).

In order to implement such approach the following techniques are used:

1. Work with workbook (portfolio) for independent tracking of subject mastering and formulation of questions. This kind of technique was used at the lectures as a form of the students' individual work organization.
2. Organization of cooperative work in groups during lectures. It was used to discuss the stated problems, to make decisions and to find of the unclear issues by the end of the day [3].
3. Individual system of practical studying that allows every student to go his own way in mastering practical skills.
4. Using during lectures forward exercises, when trainees are asked to find a solution of a problem in the process of individual work or discussions in pairs or groups before they receive knowledge on the given topic. Then a joint discussion of the problem is organized and new knowledge on the given topic is presented. After it the problem discussion and its results are analyzed.
5. Using one of basic models of active learning: *challenge-realization-reflection*.
   Challenge is a process of updating trainees' knowledge on a given topics and their motivating to investigate arisen problems.
   Realization is an acquisition of new information or ideas.
   Reflection is a phase of an introduction of new knowledge into one's own knowledge system.

For example, the first traditional straight lecture is used as a challenge. During it trainees' knowledge is updated and "areas of incomprehension" are made that will be illuminated over the training.
Challenge prepares and makes one ready for the information and the process that will be introduced at the next training works.
Realization implies input of new information. The realization of anything new is implemented only in an active learning. That is why special conditions are created to actively involve a trainee in the process of new information acquirement. Two techniques proved to be very helpful: INSERT (text marking and logging of the reading results into a special table) and "Zigzag1".
Let consider an example of "Zigzag1" application. The text of instructional lecture "*Testing of functions, the behavior of which depends upon the history of interactions between the system under test and the environment, when the factorization is required*" is split into a number of parts equal to the number of groups in the class. Before the reading whole text each group receives its own part of the text. This part should be represented in a form of a drawing or a diagram. Then each group presents their drawing to the class and answers questions. After presentations of all groups trainees get the whole text. In this case the completeness of information perception is ensured due to repeated reading of the offered text and representation of it in an another form, which is absolutely different from the initial one. This enables to involve various channels of human perception of information [10].
The following figures show two form of the same information when applying described method.

Besides, after each call of every target function with the same parameters in the same state the system should perform a transition into the same state. Groups of state dependent functions do not always meet this requirement of determinism.
Both these problems can be solved by means of state factorization, which defines an equivalence relationship on the model states. Each set (or class) of equivalent states is a generalized state. And we consider them as "states" of the system under test.
Use of generalized states instead of model states allows the test engine to reduce significantly a number of test cases and to get rid of non-determinism. Namely, it involves decrease of details in testing. The test engine tries to test each function in one of the states from each reachable set of equivalent states (i.e. in every reachable generalized state).
When we say a transition from a state to another one over a branch of a function, we mean that a call of the function in the first state brings to the branch and leads to the last state. When we say a transition to a generalized state, we mean that a transition to a state belonging to the generalized state.

Fig. 4. 1th form. The part of text offered to the group.

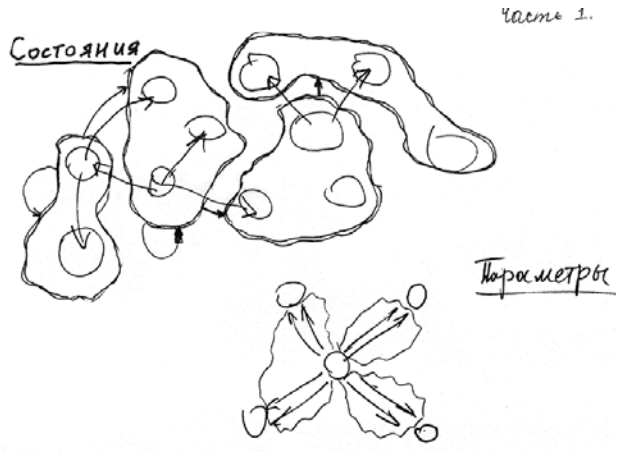

Fig. 5. 2th form. Graphical representation of the same part of text developed by the group.

Reflection is a final phase of each block of training. It includes the comprehension of activity techniques, discovery of its semantic particularities, and revelation of educational gains of the trainees.
The following reflection techniques are used:

- Verbal discussion with experts and trainees.
- Written questionnaires.
- Filling workbook.

Creation of Favorable Environment

The creation of favorable environment during training is one of the basic tasks of a trainer. This problem should be the trainer's permanent concern and the following is used to solve it:

- introductory lesson;
- daily feedback;
- adaptation of training to a particular class;
- taking into account individual learning styles [11].

The objectives of the introductory lesson are:

- getting acquainted;
- getting a picture of the class;
- presenting the course;
- acquainting trainees with the course program;
- explanation of the work with workbook;
- explanation of feedback concepts;
- creation of a favorable environment of training works.

The main component of active learning is a step-by-step performance of training practical exercises. The practical training yields success under the following psychological conditions:

- Each exercise should have a specific objective, i.e. what should be learn and how results of learning could be proved.
- Knowledge as well as clear concept of the sequence and techniques of the ordered activities to be used when performing the exercise should be represented by the material available to the trainee.
- Rules of carrying out of activities to be skilled should be acquired before the exercise performance.
- Exercises should be performed repeatedly in order to achieve higher accuracy and quickness of carrying out of activities to be skilled.
- The process and results of trainees' activities should be controlled by themselves as well as should be controlled and guided by trainers.

All these psychological conditions are created during the performance of the exercises that have to be introduced into training gradually in four stages. For this purpose, a multi-level system of exercises has been developed.

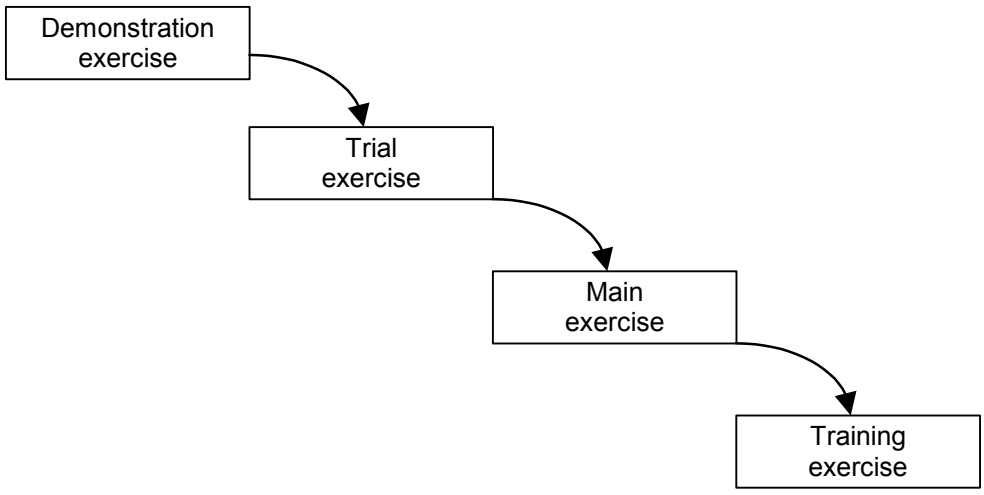

Fig. 6. Tetra-level system of practical exercises.

Demonstration exercise is carried out in order to create the clear idea of the proper sequence of actions by means of students' mental reproduction of the represented actions. It is carried out by

trainer during the instructional lecture thus initial knowledge of activities to be done is presented to the trainees.

Trial exercise is intended to acquire the right sequence of actions. After the demonstration exercise, when the trainees have the right ideas of the complete process of work, the trial exercise offers a chance for a trainee to perform it independently. Trial exercises are done until no more mistakes are made.

A trial exercise is reproductive and performed in order to acquire a *single* skill that was shown and theoretically proved in the demonstration exercise. Such an exercise may have only training purpose and could train the skill for a situation that has no practical application. When the skill is acquired it is possible to go over to the next exercise level.

Main exercise aims the use of several separate skills during its performance. It involves adding one newly acquired skill to all the preceding ones. Therefore the complexity and the volume of main exercises grows as the course evolves, because newly acquired skills are added up to the ones that have already been acquired.

Training exercise is as close to reality as possible. It is based on the application of the acquired skills that are used in various situations not always replicating trail and main exercises. Training exercise is intended to improve skills' acquirement and to train their applying under conditions of real world.

The final phase of training practical skills is the performance of ***training project***. Its objective is a development of test system for a real program. The work is done by a small group managing by one of its members.

The suggested approach was applied in the development of the following trainings:

- CTesK testing tool [12];
- J@T testing tool [13];
- J@T-C++Link testing tool [13];
- Ch@se testing tool;
- OTK tool of development of test generators for automatic testing analyzing and optimizing compiler modules.

Training participants had general knowledge of mathematics equal to an university level, C, Java or C++ programming languages knowledge. They were experienced in program development, debugging and testing.

Before the beginning of one of the trainings the participants expressed the following expectations of the course deliverables:

> "Learning testing method to the extent when it could later be further comprehended and applied"
> Dmitry M.
>
> "Take the training in testing development method. Have a chance to see what it is by completing a real project, make estimation to what extent it could be used"
> Elena S.

The final assessments of the course were:

> "The training reached the objective. After the training I realize how and where the testing tools could be used very well"
> Dmitry M.
>
> "I enjoyed the course. The result is that I received a certain ideas about testing methods. The method is interesting. It is worthwhile to deploy and apply it at enterprises"
> Elena S.

Conclusion

The described methods have been applied for development of training courses in tools CTesK [12], J@T and J@T-C++ link [13], Ch@se, OTK. The trainings were conducted in

- Lanit-Tercom (Russia),
- Systematic Software Engineering (Denmark),
- GosNIIAS (Russia),
- Saarland University (Germany),
- ATS (India),
- Luxoft (Russia),
- Intel Technologies Inc. (Russia),

as well as in research groups of ISP RAS and at the Moscow State University.

The experience shows that the most trainees completely copes with the tasks and becomes so proficient in the application of the tools that they can use them without expert's assistance just after training.

The staring phase of training still represents a difficulty yet to be overcome. During that phase the class has to be prepared for active learning, which requires a lot of effort from those who are accustomed to academic way of learning. The training duration is yet another problem. Even though it was possible to reduce the training period from 2 - 3 months to 4 - 5 days, it is still considered as too long for the many commercial companies. The possible way out of these predicaments is to split the training into several parts and to separate special blocks that address managers, architects, developers, testers, etc.

Learning methods developed for training can be applied in university courses. Full-scale application of active and cooperative learning in large classrooms is difficult, because it requires group discussions, speaking individual opinions by each student, project proving. But practical studies can be completely taken from industrial training courses. It provides quick and well acquirement of knowledge and practical skills. Also such practical studying teaches the students to make independent decisions based on an analysis of various factors.

The experience obtained by RedVerst in communication with potential and actual UniTesK users shows that the successful transition of advanced development methods requires increased attention to organizational and cultural aspects of their use in the industry. This is especially true for the ones based on formal methods or other techniques usually accessible only for people with university education in mathematics and strong skills in manipulating abstract objects.

Our experience based on interaction with software developers and test developers in various software development organization approved that it is necessary to use different learning methods to solve different problems of technology transition. Traditional academic courses are suitable to train the future staff, if they are backed by the practical tasks using real software examples. Training courses based on active learning and immersion in the learning process are suitable to train end users off the tools supporting the technology. Special trainings are required for project managers. Such training should give skills useful for adequate management and control of projects using the new technologies. In particular, new metrics should be given to measure the state of development process and quality of the resulting software. Besides all the previous, special short-term trainings, seminars, presentations, textbooks and manuals are needed for wide popularization of advanced development methods and preparation of their adequate transition to the industry.

A lot of problems of learning preparation are not solved already — some of them are related to choosing an appropriate learning method, others concern specialized training for managers of various levels. Nevertheless, the approach based on adequate education was approved as very suitable to solve organizational and cultural problems of advanced development method transition to the industry. We also have no doubts that this success is not related to specifics of UniTesK technology considered as an example of such a method.

References

1. http://www.ispras.ru/groups/rv/rv.html
2. The RAISE Language Group. The RAISE Specification Language. Prentice Hall Europe, 1992.
3. V. Kuliamin, A. Petrenko, A. Kossatchev, I. Burdonov. UniTesK approach to test development. Proramming and Computer Software, 29(6), 2003.
4. A. Barantsev, I. Burdonov, A. Demakov, S. Zelenov, A. Kossatchev, V. Kuliamin, V. Omeltchenko, N. Pakoulin, A. Petrenko. UniTesK Approach to Test Development: achievements and Prospects. Proceedings of ISP RAS, No. 5, 2004.
5. http://www.ispras.ru/groups/rv/tutorial.html
6. Khutorskoi A. Modern didactics. In Russian, Piter, 2001.
7. Pedagogics. Pedagogic theories, systems, and technology. S. A. Smirnov, editor. In Russian. Moscow, 2000.
8. J. Hartley and I. K. Davies. Note-taking: A critical review. Programmed Learning and Educational Technology, 15, 207-224 (1978).
9. L. Rae. Using Activities in Training and Development. Kogan Page, 2000.
10. Petrenko O. L., Prokophieva L. B. and others. Technologies of open learning. In Russian, Moscow, 2002.
11. K. Thorne, D. Mackey. Everything You Ever Needed to Know About Training. Kogan Page, 1996.
12. http://www.unitesk.com
13. http://www.atssoft.com