Rapid Training on Specification Based Testing Tools

Olga L. Petrenko, Vitaly A. Omelchenko

Institute for System Programming of Russian Academy of Sciences (ISPRAS),
B. Communisticheskaya, 25, Moscow, Russia
{olga, vitaliy,}@ispras.ru
http://www.ispras.ru/groups/rv/rv.html

Abstract. The article describes the procedure of training in the advanced testing technology supported by UniTesK test development tools. It considers both the general problems of development of trainings in novel technologies, and specifics of formal software development methods application in practice. The related problems of university education are also discussed.

1 Introduction

One of the reasons that obstruct introduction of sophisticated technologies including advanced testing technologies is the problem of personnel training. If a period of time necessary for the user to learn is too long it drastically reduces the economic efficiency of the new technology. Also the efficiency of utilization of a new technology depends on the degree to which personnel wields not only the technology as a whole but also separate instruments that support the technology. This article closes up on the problems of training in the use of separate instruments, even though the technologies and even testing methods are also partially touched upon.

The article presents a nearly decade's wealth of users' training experience in advanced testing technologies and, to be more precise, on the technologies based on formal specifications. Experiments were carried out at university courses, at industrial personnel training courses and in research groups. Leaping ahead we would like to point out that attempts to use the traditional university training methods at industrial personnel training courses yielded negative results. On the contrary, the attempt to transfer the methods of industrial trainings to university lectures and workshops produced positive results.

The article is composed as a description of the process of training development and its implementation in one of the testing tools of UniTesK family.

UniTesK is a testing methodology based on formal specifications. All the UniTesK family tools utilize common functionality description methods (specifications) and standard testing system architecture. The tools differ from each other in programming language platforms, for which they are designed, which are C, C++, Java, C#.

Educative training was chosen as a way to teach UniTesK testing tools. Educative training is considered as the basic course, the knowledge of which enables a person to test programs. On the other hand, it represents the basis, which allows further perfection of testing mastership and to use the co-verification process properly. The co-verification process [6] is the software development process, which involves the development of facilities of software correctness verification concurrently with the development of the software itself. This is a promising approach to the high quality software production.

The objective of the training is the development of knowledge, skills and attitudes for design of tests based on specifications. The training offers a ready technology that consists of the following phases:

1. Specification of the target software system — the formal description of software requirements.
2. Development of test scenario — a compact test description defining a sequence of impacts on the system under test.
3. Development of mediators — test system components binding the specifications with the implementation under test. They allow more abstract and comprehensible specifications.
4. Test system debugging.
5. Test result analysis.

The following principles were used as the foundation of the educative training in order to obtain quick and high-quality mastering of the mentioned above skills:

1. Thorough selection of training content in accordance with the objectives set;
2. The choice of adequate training forms;
3. Training of the students in active learning mode — an approach to learning that encourages inquiry and discovery;
4. Establishment of favorable educative environment [1].

Each of these principles is implemented in a certain way in the training offered. The implementation of these principles made it possible to cut the training period to 4-5 days and grant the training results.

2 Selection of Training Content

The training content is the system of knowledge, skills and attitudes that have to be mastered and the experience of the creative activity.

Formation of the training contents is based on the following principles:

- Consideration of the social demand;
- Compliance of the training contents to the objective set;
- Uniformity of the contents, activities and regulations of education process;
- Apprehensibility of the training contents.

The principle of training material generalization is used for content selection during which one or several basic principles are distinguished. These selected principles are later comprehensively and repeatedly elicited using various approaches. In our case the following main features of UniTesK may serve as such principles:

- Testing is considered as a comparison of behavior features of the implementation to model features that is determined by its own specifications;
- Implementation is considered to be a set of interfaces (operations, procedures, data structures, etc.). The internal structure of implementation and its algorithms is not discussed, i.e. "black box" approach is used;

- The specification is given in the "implicit" form, in the form of preconditions and postconditions of the operations and invariants of data structures;
- Implementation and model interfaces could differ both in structure and in the abstraction level; connection between the correspondent interfaces is assigned by special mediator programs;
- UniTesK offers a unified architecture of the test system, which implies nodes for test impact generation for analysis of system behavior under test as well;
- Subsystem of test impact generation is split in two levels — generation of input test data for separate operations (iterators of values) and generation of operation calls sequence (test sequence);
- The idea of a FSM traversal that is constructed from specifications of operations and test scenario underlies the bases of the test sequence generation method.

The training contents selected in compliance with these principles is represented in a form of a training program. Fig. 1 demonstrates the example of a program of the developed training in CTesK-lite testing technology.

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 interaction of the system with the environment:
   - Definition of function precondition;
   - Definition of functional branches;
   - Definition of function postcondition;
   - Definition of type invariants;
   - Definition of test scenario;
   - Definition of parameter iteration.
2. Testing of functions, the behavior of which depends on the history of interaction of the system with the environment, and which do not require generalization:
   - Definition of function precondition;
   - Definition of functional branches;
   - Definition of function postcondition;
   - Definition of type invariants;
   - Definition of invariants of state variables;
   - Definition of test scenario;
   - Definition of parameter iteration.
3. Testing of functions, the behavior of which depends on the history of interaction of the system with the environment, and which require generalization:
   - Definition of function precondition;
   - Definition of functional branches;
   - Definition of function postcondition;
   - Definition of type invariants;
   - Definition of invariants of state variables;
   - Definition of test scenario;
   - Definition of generalized state;
   - Definition of parameter iteration;
   - Transformation of generalized parameters to function ones.

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). ”Opened” 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. 1. A program of the training of CTesK-lite testing technology

3 The Choice of Adequate Training Forms

The immersion method is chosen as the principle training form. This means that students are mastering only one testing tool during the complete training course, which takes 4 to 5 days.

The training foresees the following forms of classes conduction:

1. *A traditional lecture* assumes that a lecturer works with the whole class using the same pace and general tasks. This form is used to present new theoretical material.

The theoretical material is used in instructive lectures, which directly precede the workshops.

2. *An instructive lecture* familiarizes the students with the technology of the activity in offing, with particularities of carrying out of certain actions, techniques, algorithms of problem solving. Taking into consideration the complexity of the course studied, the main notions are defined in the instructive lectures as well as the structure of notions of the selected part of a knowledge domain and the links between them.
3. *Workshops* are carried out after significant parts of training have been studied. They take shape of practical assignments that are performed independently.

   In the given course the practical assignments are specification writing, development of test scenario, running and debugging of tests.

   The first method of controlling students' activity in the workshops is the independent choice of a sequence of stages in problem solving and self-rating of how it was performed.

   The sequence of stages could be compiled from a set from offered ones, but not from arbitrary ones.

   The set of stages offered is as follows:
   - Familiarization with a task
   - Independent performance of the task
   - Comparison of the performed task to the key
   - Study of an offered task performance algorithm
   - Familiarization with performance of one stage of the offered algorithm

   In order to carry out a certain task various kinds of stage sequences could be compiled.

   Sequence 1:
   - Familiarization with the task
   - Independent performance of the task
   - Comparison of the task performed with the key

   Sequence 2:
   - Familiarization with the task
   - Study of an offered task performance algorithm
   - Independent performance of the task
   - Comparison of the task performed with the key

   Sequence 3:
   - Familiarization with the task
   - Study of an offered task performance algorithm
   - Familiarization with performance of one stage of the offered algorithm
   - Independent performance of the task
   - Familiarization with performance of one stage of the offered algorithm
   - Independent performance of the task and etc.
   - Comparison of the task performed with the key

   The second method of controlling students' activity in the workshops is the performance of special tasks.

4 *Consultations of specialists* (individual kind of training) is used as an auxiliary training option aimed at clarification of certain problems as well as at more profound mastering of the subject.

4 Students' Training in Active Learning Mode

When choosing the training technique it was considered that the training participants take part in various projects at work, which require creative thinking and social activity. That's why it was decided to create familiar and natural environment for work.

When active learning mode is used students 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 [2]. It is well known that when active learning mode is used 75% to 90% of the material is mastered. (Compare: Traditional lecture yields mastering of 5% of the material introduced in it).

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

1. Work with portfolio (workbook) 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 group work at the lectures. It was used to discuss the stated problems, to make decisions and to defining of the unclarified problems by the end of the day [3].
3. Individual system of practical workshops that allows every student to go his own way in mastering of a part of the course.
4. Use of advanced tasks at the lectures, when the students are asked to find a solution of a problem in the process of individual work or in pairs or in subgroup discussions prior to receiving the bulk training material on the given topic. Organization of discussions about the offered subject and subsequent analysis of the discussion process after acquisition of new material.
5. Application of one of the frameworks of active training: evocation, realization of meaning, reflection.

   Evocation is the process of knowledge updating on the given problem and creation of curiosity to its investigation.

   Realization is an acquisition of new information or ideas.

   Reflection is the stage when the new knowledge is introduced into one's own knowledge system.

For example, the first traditional lecture is used as an evocation. In such a lecture, the possessed knowledge is updated and "areas of incomprehension" are formed that will be liquidated in the process of the subsequent training.

An evocation prepares and makes one ready for the information and the process that will be set forth at the next stages of work.

The stage of 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 student in the process of fresh information mastering. Two techniques proved to be very helpful: INSERT (text marking and logging of the reading results into a special table) and "Zigzag1".

Let us consider an example of "Zigzag1" application. The text of instructive lecture "*Testing of functions, the behavior of which depends upon the history of system interaction with the environment with generalization*" is split into a number of parts equal to the number of subgroups in the class. Each group receives an abstract of the text (see Fig. 2) and the task to represent it in a form of a drawing or a diagram (see Fig. 3). Then each group presents their drawing to the class and answers questions.

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. 2. An abstract of the text that was offered to a group

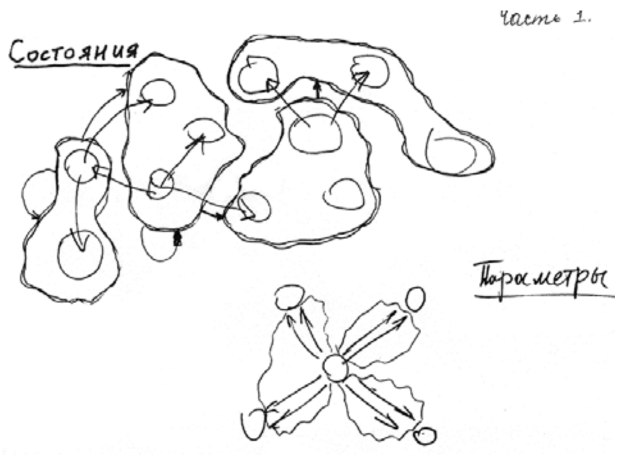

Fig. 3. The graphical representation made by the group of trainees

After all presentations have been made, everyone in the class receives the complete text of the lecture. The comprehensiveness of information perception in this case is ensured due to repeated reading of the offered text and due to representation of it in another form (which is absolutely different from the initial one). This enables to involve various channels of information perception in a human being [5].

Reflection is the final stage in each block. Reflection is the comprehension of activity techniques, discovery of its semantic particularities, revealing of educational gains of the students.

The following reflection forms were used:

- Verbal discussion with experts and students
- Written questionnaires
- Filling in of portfolio

5 Creation of Favorable Training Environment

The creation of favorable training environment 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;
- Consideration of individual learning styles [4].

The objectives of the introductory lesson are:

- Getting acquainted;
- Getting an impression of the class;
- Presentation of the course;
- Familiarization with the course program;
- Explanation of the work with portfolio;
- Explanation of feedback principles;
- Creation of a favorable working environment

The main component of active technology mastering is a step-by-step performance of training tasks. For this purpose, a multi-level system of exercises has been developed.

The exercises yield success under the following psychological conditions:

- Determination of a specific objective for each exercise, i.e. what should be mastered and to what extent of accuracy
- Not only knowledge but also clear concept of the sequence and techniques of the impending action performance that should be mastered in the exercise
- Preliminary mastering of the rules of performance of the activities mastered
- Repeated performance of activities being mastered in order to reach higher accuracy and quickness
- Permanent self-control of the process and results of the activities as well as the control and guidance of the teacher

All these psychological conditions are created during the performance of the exercises that have to be introduced into training gradually in four stages.

A familiarization exercise is carried out in order to create a clear idea of the proper sequence of actions with the help of students' mental reproduction of the impending actions. The familiarization exercise is carried out by the trainer during the instructive lecture while presenting the initial information on the work to be done.

A trial exercise is carried out to master the right sequence of actions. After the familiarization exercise, when the students acquire the right idea about the complete process of work, this exercise offers a chance for a student to perform it independently. Trial exercises are done until no more mistakes are made.

A trial exercise is reproductive and is performed in order to master a single skill that was shown and theoretically proved in the familiarization task. Such a task may only be of training purpose and could master the skill for a situation that has no practical application. When a skill is mastered it is possible to go over to the next exercise level.

Main exercise is carried out in order to engage several separate skills to do a job. A main exercise involves adding one more newly acquired skill to all the preceding ones. Due to that, the complexity and the volume of main exercises grows as the course unfolds, because newly acquired skills are added up to the ones that have already been mastered.

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 educational exercises. Training exercise is used to master skills and to develop the abilities for the performance of a task in real conditions.

The final stage of work is the completion of a *training project*. The objective of the project is to write a testing system for a real program. The work is done by a subgroup under the guidance of one of the members of the group.

Let us consider examples of various kinds of exercises performed during the study "*Testing of functions, the behavior of which does not depend upon the history of system interaction with the environment*".

Fig. 4 presents the familiarization exercise performed by a trainer during this instructive lecture.

Task:
Develop a test for testing of a function substituting a character in a string
`char replace char by index(char* str, char new, int index)`
Function description:
Inputs:
- A string of non-zero length (`char* str`), containing only `a-z`, `A-Z` and `0-9`;
- index (`int index`), of a symbol to be substituted, which should be greater than or equal to `0` and less then the length of the string;
- new character (`char new`) should be `a-z`, `A-Z`, `0-9` or string end character.

Results:
- After the function is called the character value in the string by the passed index coincides with the value of the new character;
- if the value of the new character does not coincide with the value of the character being substituted the function returns the old character value in the string by the passed index, otherwise the end of string character is returned;
- if the function receives zero pointer it returns end of string character.

Fig. 4. Example of familiarization exercise performed by a trainer during the instructive lecture

When performing this exercise the following skills are demonstrated:
- Definition of type invariants

– Definition of access constraints of parameters
– Definition of constraints of parameter values in precondition
– Definition of functionality branches in accordance with parameter values
– Definition of constraints of parameter and result values in postcondition
– Use of parameter pre-values in postcondition
– Definition of transformation of parameters and results in mediators
– Development of test scenario with single state

Fig. 5 presents the trial exercise. Performing it trainees apply in practice a one of the skills represented during the instructive lecture — "Definition of functionality branches in accordance with parameter values".

Task:
Define functionality branches of the function and describe them in the coverage block of specification function

```
specification int f(int a, int b, bool neg)
reads (int)a, (int) b, (bool) neg;
```

Function description:
The function receives two input integer values and one logical value. If the `neg` parameter value is false the result of the function call must satisfy the following rules:

- if `a` and `b` are of the same sign, then the function must return their sum
- if `a` and `b` have different signs, then the function must return their difference
- if `a` is equal to zero, then the function must return zero
- if `neg` is true, then only the result sign must differ from the cases above.

Fig. 5. Example of trial exercise for teaching one skill

Fig. 6 presents the main exercise. The exercise requires applying a set of skills represented during the instructive lecture.

Task:
Write the specification and mediator for a function

```
int se(double a, double b, double c, double* x1, double* x2)
```

that solves quadric equation. For the types on input parameters `x1` and `x2` determine invariants and consider them in writing of specification.
Function description:
The function receives input values of `a`, `b` and `c` which are the coefficients of quadric equation. Parameter `a` must be non-zero. As a result the function returns the number of equation roots and stores the values of the roots into non-zero pointers `x1` and `x2`. If the equation has only one root the values stored by the pointers should be equal. If there are no roots then the values stored by the pointers are not determined. Floating point arithmetic has unlimited precision.

Fig. 6. Example of main exercise that requires applying a set of skills

When performing this task the following previously acquired skills will be needed:
– Definition of type invariants

– Definition of access constraints of parameters
– Definition of constraints of parameter values in precondition
– Definition of functionality branches in accordance with parameter values
– Definition of constraints of parameter and result values in postcondition
– Use of parameter pre-values in postcondition
– Definition of transformation of parameters and results in mediators

Fig. 7 presents the training exercise. The task is performed at the end of training and includes all the skills mentioned above and other skills mastered during the training.

Task:
Develop a testing system (a test) for memory management system functions testing.

System description:
The system implements the functions of memory allocation, memory release and resizing of memory blocks. Memory blocks do not overlap. Interface functions:

```
        void* calloc(size t nmemb, size t size);
        void* malloc(size t size);
        void free(void* ptr);
        void* realloc(void* ptr, size t size);
```

Function descriptions:
`calloc()` allocates memory for an array of nmemb elements each of size size and returns the pointer to this memory. The allocated memory is populated with zeroes.
`malloc()` allocates size bytes of memory and returns the pointer to the allocated memory. The allocated memory content is not cleaned.
`free()` releases memory at which ptr points, which has been previously returned by calling `malloc()`, `calloc()` or `realloc()`. Otherwise of in case when `free(ptr)` has already been called, the behavior is not determined. In case of zero ptr no action is taken.
`realloc()` resizes a memory block, at which `ptr` points in such a way that its size becomes equal to size bytes. The memory contents by the pointer does not change; newly allocated memory is not initialized. A call with `ptr` equal to `NULL` is equivalent to calling of `malloc(size)`. A call with size equal to zero is equivalent to a call of `free(ptr)`. If `ptr` is not equal to `NULL` then it must be returned by the previously called functions `malloc()`, `calloc()` or `realloc()`.
Returned values:
For `calloc()` and `malloc()` the returned value is a pointer at the allocated memory or `NULL` if it wasn't possible to allocate memory of the necessary size. In case of zero size allocation attempt the memory is still allocated.
`free()` returns no values.
`realloc()` returns a pointer at the allocated memory which can differ from `ptr`, or `NULL`, if memory allocation of the necessary size failed or size was equal to zero. If memory allocation failed the initial memory block remains intact, it is neither released nor moved. If there is an attempt to allocate zero memory size (`realloc(NULL,0)`) the memory is still allocated.

Fig. 7. Example of training exercise is performed at the end of training

The suggested approach was applied in the development of the following trainings:
– CTesK testing technology;
– J@T testing technology;

– J@T-C++Link testing technology;
– Test generation creating technologies for automatic testing of analyzing and optimizing compiler modules.

The training participants had general knowledge of mathematics equal to an university level, C or Java 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:

"*Mastering of testing technology to the extent when it could later be further comprehended and applied*"
Michael S.

"*Take the testing technology course. Have a chance to see what it is by completing a real project, make an estimation to what extent it could be used*"
Elena S.

The final assessment of the course:

"*The training reached the objective. After the training I realize how and where the testing tools could be used very well*"
Michael S.

"*I enjoyed the course. The result is that I received a certain idea about testing technologies. The technology is interesting. It is worthwhile to deploy and apply it at enterprises*"
Elena S.

6 Conclusion Students' Training in Active Learning Mode

The described methods have been applied for development of trainings in tools CTesK [7], J@T and J@T-C++ link [8, 9]. The trainings took place at Lanit-Tercom (Russia), Systematic Software Engineering (Denmark), GosNIIAS (Russia), Saarland University (Germany), ATS (India), as well as in research groups of ISP RAS and at Moscow State University.

The experience shows that a large number of students completely copes with the tasks and becomes so proficient in the application of the tools that they can use them independently right after training practically without any consultations with experts.

The beginning phase of training could be mentioned as a stage, which still represents a difficulty yet to be overcome. During that phase the class has to be prepared for active learning methods, which requires a lot of effort from many of those who are used to academic way of learning. The training duration represents 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 to be too long for the majority of commercial companies. The possible way out of this predicament lies in splitting of training into several parts and in distinguishing of special blocks that address managers, architects, developers, testers, etc.

In the modification of J@T training for university students, a combination of lectures (with active methods reduced to a minimum) and workshops that were adopted from the training devised for industrial enterprise was used. The experience proved to be very successful. The complete course consisted of 3 lectures, 3 workshops and 1 test, though, of course, the volume of training material and of acquired skills in the university course was considerably reduced.

References

1. Khutorskoy A.V., Modern didactics (in Russian). Saint-Pitersburg, 2001.
2. Smirnov S.A., Pedagogics. Pedagogic theory, systems, technologies (in Russian). Smirnov S.A., Moscow, 2000
3. Lesly Ray. Exercises: schemes and strategies (in Russian). Translation from English. Saint-Pitersburg, 2003.
4. Kay Thorn, David McKay. Training Trainer's handbook (in Russian). Translation from English. Saint-Pitersburg, 2001.
5. Petrenko O.L. Prokofieva L.B. et al. Technologies of the open education. Moscow, 2002.
6. http://www.ispras.ru/groups/rv/tutorial.html
7. http://unitesk.ispras.ru
8. http://jatt.ispras.ru
9. http://www.atssoft.com