Use of Non-IT Testers in Software Development

Vineta Arnicane

University of Latvia, Raina blvd. 19, Riga, Latvia
vineta.arnicane@lu.lv

Abstract. Because of a shortage of IT specialists, many companies which are not involved in software development business are forced to use employees who have minimal or no any knowledge about software development as testers. The author of this paper has used years of experience in working with such testers to provide a description of them, looking also at their most typical testing styles and the problems which occur for testers, their colleagues and bosses, and the overall software development processes. Non-IT testers often feel like second-class employees, because they are forced to work in an environment in which they do not have sufficient skills. This paper reviews issues which should be taken into account when training these testers, including the question of what exactly they should be taught. Examples and conclusions are used to provide advice on the more effective use of non-IT testers to achieve positive results.

Keywords: Software testing, testing process improvement.

Introduction

In recent years, greater and greater attention in the world of software development has been devoted to testing. At higher education institutions, testing-related training has not attracted sufficient focus, however. Software design companies usually try to deal with the issue of training testers on their own. Various educational programs have been designed by software development companies. Alternatively, training services are offered by specialized organizations or companies.

Companies at which software design is not the basic area of operations lack professional testers. Often they lack well-qualified IT specialists as such. Many companies hold on to the myth which says that testing is easy and uncomplicated, that it can be handled by any employee [1]. Even software development companies sometimes use beginners in the field as testers.

Non-IT companies often bring in people who know nothing about testing and, in some cases, nothing about IT as such. When testing does not yield the expected results, companies find themselves ready to pay for the training of these individuals in the field of testing.

This article is based on the author's more than 10 years of experience in various jobs in the industry. She has worked with non-IT testers and dealt with their training. This paper reviews the most important observations and conclusions vis-à-vis issues

such as things which non-IT testers understand intuitively and without training, the things about testing which they must be taught, the testing methods which they are able to comprehend and apply, and the way in which non-IT testers should be managed.

The author of the paper has conducted several training sessions for non-IT testers, both informally and individually, and by developing training courses and teaching specialized lessons for client organizations (the most detailed course involved 40 hours of study).

Non-IT testers

People who are used as testers by companies can have different levels of knowledge in the area of IT and testing, as well as in the area of the relevant company's business. Companies lack professional testers, and it is hard to find people with good IT skills to work as testers – even if they have little or no knowledge about the software development process as such. This means that companies are forced to turn to people who know about the company's business but have weak or no knowledge about testing or IT as such. In this article, let us refer to them as non-IT people or non-IT testers. People with IT knowledge will be defined as IT people or IT testers.

Non-IT people engage in testing on the basis of various circumstances. Sometimes they are given full-time jobs at company IT departments as testers. Other times testing is just a part of a broader set of duties.

The advantage of non-IT people is that they can have in-depth knowledge about the area of business for which IT software is being designed. Such employees often have years of experience in the relevant area, but they have little knowledge about the technologies that are used in system development and the architecture of the resulting system. They don't know much about testing methods and cannot prepare good reports on problems that have been identified.

The intuitive testing style of non-IT testers

If non-IT people are not trained as testers, they do the work intuitively. There are several characteristics which different testing styles have in common. Let us review the most important aspects of the work of non-IT testers:

- **They use functional testing**: Non-IT testers are more likely to engage in ad hoc testing – without any system or consistency, often evaluating a system just from the visual perspective, testing standard usage, choosing the "right path" with the right data, and doing work in the order in which it is usually done;
- **They are afraid of destroying the system**: Perhaps intuition or experience tell a non-IT person that the work of software could be damaged, but the person does not do this so as not to create problems for colleagues and to allow them to continue their work and to finish the development project on time. Potential problems are not examined or analyzed, and the necessary problem reports are not filed;

- **They consider a test case to be only a series of operations, not the relevant data**: Data planning, if any, is too primitive. All input data are sometimes seen as identical, as factors which cannot really change testing results or procedures, so the data which are entered and used during the testing process are not recorded. The results are often evaluated casually – "that's approximately how things should be";
- **They intuitively use boundary values and partitioning of data in equivalence classes**: If attention is focused on test data, border instances are intuitively examined. This testing method is used at a fairly normal level;
- **They cannot work with a database**: Testers usually cannot prepare data in a database, read data from tables, determine data which are not shown in user interfaces or LOG data, or they cannot test whether the information on screen or in a report is in line with data in the database;
- **In testing, they use test data which are close to reality, establishing typical situations and chains of activities**: This is a great advantage for non-IT testers if they come from a business with business knowledge. They know HOW and WHERE the system will be used. On the other hand, they find it unnecessary to test non-standard applications of the software, assuming that users know what they are doing;
- **They do not support testing as early as possible**: These people do not see the need for early testing, and they have problems in establishing testing examples on the basis of requirements. It is hard for them to establish and examine a potential system. Non-IT persons see this exclusively as the testing of ready-made software – i.e., they need a program which can be really executed with test cases. They think that systems must be tested and tests must be prepared only when the software is ready, when it can be seen and used in real terms;
- **Sometimes they have stereotypical thinking**: If an organization has already used an IT system, that puts a stamp on thinking about the business organization and about how the system should appear and what it should do. It is hard to ensure a fundamental shift in approach, even though this is occasionally needed when new technologies or necessary changes in business processes are implemented;
- **They have problems in noting and describing problems**: Non-IT testers do not react to "odd" behavior in the system even if it happens repeatedly, pointing to various secondary factors which might have created the problem. They do not try to ensure a repeat of the questionable situation on purpose;
- **They are not able to prepare good problem reports**: Usually there are problems in preparing problem reports which go to the heart of the problem, which are clear, precise and brief. Very rarely does anyone check the report to see whether the software designer will be able to find mistakes on the basis of what has been reported;
- **They do not know how to document their work**: Testers not only do not know how to document what they are doing, but they are sometimes afraid to do so, because they do not feel sufficiently competent. Sometimes they do not want to spend time on documentation, because they consider that to be a pointless waste of resources;
- **Testing is led by IT people**: Non-IT testers feel insecure, they do not defend their judgments in those cases when requirements are incomplete (e.g., when there are

no requirements as to usability or performance). If system developers for one reason or another do not want any official registration of a problem, then they insist that the discovered problem is not a mistake and must not be registered so as to improve statistics or to put an official end to a stage in design. This often means that problems are not registered even though they are very important, albeit not readily visible. The result is that fundamental system aspects do not attract sufficient attention, and sometimes they are not tested at all. Companies make peace with more or less useful software that has been developed by the system developers, instead of insisting on their own views and ensuring that problems are recorded in reports or demands for change. Testers wait until real users report a similar problem before reporting on other, identical problems.

Non-IT testers have problems with situation in which the knowledge of a software development or professional tester is needed about issues such as:

1. The way to test a real business situation – long-lasting use of a system, changing of the server time and date to the end of the month or year;
2. The way to select test examples so that the testing is sufficiently effective, using one test example to test various aspects of the system (introducing the synchronization point);
3. The way to fill in a database with test data if the system does not have an appropriate user interface for data entry or if the data come from outer interfaces in the system;
4. The way to prepare larger volumes of testing data;
5. The way to prepare, maintain or commission test beds;
6. The way to test external interfaces.

Non-IT testers are happy to learn about testing procedures if the process is not too complex or long-lasting, but there are problems with training materials. People without a grounding in IT can find it difficult to use existing books or other sources related to testing, because these usually require knowledge about computer sciences and, particularly basic software design skills.

On the other hand, companies wish to train non-IT people to do testing quickly so as to achieve better results. This means a search for people who can adapt training materials to company needs and can provide special training.

Training of non-IT people for software testing

These are the things which non-IT people hope for when it comes time for training in the area of testing:

1. There will be no “technical details” – bytes, bits and symbol codes shouldn’t be mentioned at all;
2. There will be no need to study the software development methodology – that is a problem for the bosses, we will do what we are told to do;
3. The process will be as close as possible to the would-be used working style – “we already do all our work properly”;

4. There will be as many examples as possible from the system and situation that are to be tested so that the method can be learned more easily and put to use;
5. There will be nothing complicated even that seems to be the case at start – no techniques that is hard to be understood, even if they are effective in use;
6. There will be no requirements related to anything which has to do with programming or the technologies of software development.

What can be taught to non-IT people when it comes to testing? What are the skills and areas of knowledge which can be learned comparatively easily? What will allow such people to make use of their strengths – their high level of knowledge about the relevant business?

- **Defining uses on the basis of requirements**: Non-IT testers can make use of their extensive knowledge about their business and its processes and dependencies. Of use here is knowledge in the drafting of models and the use of cause-and-effect diagrams [3] so as to justify and document the testing scenario which has been chosen;
- **Use of boundary values and equivalence classes in the selection of testing data**: Intuitive methods can be improved with additional knowledge, or else testers who are unfamiliar with these methods can simply be given a powerful weapon in terms of coming up with good testing examples;
- **Methods to reduce the number of test examples**: Here we refer to knowledge about screen form elements that are grouped according to dependencies, to the use of orthogonal arrays, decision tables and analysis of combinations;
- **Recording of test examples**: People must learn how to record the test, the entered data and the expected results;
- **Knowledge about usability requirements**: Non-IT people should be taught about desirable placement of elements, the appearance of screen forms and reports, undesirable practices and common mistakes. This knowledge allows testers to have a greater belief in themselves, it allows them to justify their views and to stand up against the pressure of system developers. Sometimes companies have their own guidelines as to the appearance of user interfaces, and then testers can look to see whether the guidelines are of an acceptable level of quality;
- **Risk analysis**: Risk analysis [2] makes it possible to organize work more effectively under conditions of limited time and resources, testing the most important requirements and engaging in the most critical tests. Sometimes testers do not want to do this, arguing that risk evaluation requires too much bureaucracy;
- **Basic skills in SQL queries**: These allow testers to look at data in a database. In more complicated cases, it is better if the query is written by someone from the development group, but testers must know how to use the queries as necessary. Queries of this kind, for instance, allow testers to check the LOG records of the system;
- **The role of the tester in projects and the aims of testing**: Testers have a better sense of their job of finding mistakes than of proving that the system is faultless. They understand their capabilities in terms of those mistakes which they can and cannot define. Testers must understand that they alone cannot ensure the quality of software, and they are not responsible for any shoddiness on the part of the software developers [4];

- **Writing problem reports**: Testers must know how to prepare such reports, and they must be aware of the required content [5]. It is desirable to remind them of simple methods to note problems – how to obtain a snapshot of the screen, how to use graphics editing to cut the necessary part out of an image, how to place the image in the problem report itself, etc.;
- **Real examples**: Training should be based on examples from the system which is to be tested – requirements, screen forms, reports, etc. In that case the training uses concepts and business terms with which testers are familiar. It is easier to absorb and remember new knowledge. Even if there is use of materials related to a part of the system which has been tested and approved for use, people are sometimes surprised at how many problems of that system part have gone unnoticed.

Specifics in managing the work of non-IT testers

All testers have many management problems in common, irrespective of how much they know about IT. Management of non-IT testers, however, can be a specific situation, largely as a result of the fact that non-IT testers feel greater humility is dealing with system developers. That is because they feel insecure about their own competence in the field of IT.

- **Testers must feel that they are authorities and that their work is necessary**: During training, there is often revelation of things which testers do not dare say on an everyday basis in conversation with their bosses – that there is insufficient interest in their work, that they feel abandoned, that the results of their work are ignored, that instructions have been given to define or not to define a problem, etc. No one is interested in the fact that these people know or do not know something specific about their work. Testers don't know where to go for help. Training of testers, therefore, should be offered in the presence of each tester's immediate superior;
- **Managers must monitor the time use of the tester**: Non-IT testers will need additional time to design tests, at least at first time period. If there is insufficient time, then testing will be limited to functionality in relation to standard usage. Eventually such a tester will start to believe that there has been enough testing. When repairs of mistakes are investigated, for instance, only the test which found the problem is carried out, and there is no thought to the possible side-effects to the fix;
- **It is dangerous if non-IT testers are subordinated to the leader of the developer group**: Non-IT testers often yield before the views of IT people without insisting upon their own experience and views about what the system should be like. If the leader of the group is not interested in learning about mistakes for one reason or another, then the tester is told that the mistakes are not mistakes, that no problem report must be written and that the mistakes do not have to be registered. For instance, there is no acceptance of mistakes related to the usability of software – ones that are the result of the poor quality of the system's design and ones that are expensive to fix now. The consequences is that the tester

calmly waits until users report the problems, because group leaders usually respect the views of users;

- **System developers and testers must both establish internal problem registers**: If there are artificial limitations on the list of reported problems, then both developers and testers need their own internal problem registers, usually in the form of spreadsheets. Testers can use the register to remember those cases when there were suspicions of a mistake, but it was not possible to repeat or to localize it sufficiently to prepare a problem report. If such suspicious cases are kept in mind, it is possible to reveal the mistakes more effectively later, when they have appeared again;
- **A dictionary is necessary**: Software developers and testers from the business side of the operation must speak the same language and use the same contexts in the same sense. There must be agreement on terminology. In practice it has been found that there is great chaos in the use of testing terms.

Case studies

Let us now look at three cases in which non-IT testers became involved in software testing.

Case I

At the very beginning of the project, one or two authorized users were defined in each department of the organization to provide consultations to system developers as the software was being created. User representatives had access to the testing environment, and they were ordered to test all of the operations which they might use in their work. The user representatives were trained to use the system, but they were not trained in the area of testing.

Initially there was little use of the system. Statistics about the number of log-ins and the amount of time spent in the system were distributed, and then testers tried to hook up to the system. After reviewing it a bit, they stopped the work. They did not log out of the system, however, hoping to improve statistics as to the amount of time that had been spent in the system.

When statistics were made available about the screen form used in the system, the number of reports and the time when these are presented, true work in the system began, but not in a planned way. The authorized users tested the functionality when requested to do so by the system developers.

The first levels of testing were handled by the development group. Non-IT testers were more involved in the stage of accepting the results. For few working days they handled as many of their everyday operations as possible in the testing environment, and they also tested the operations which must be handled at the end of a month or a year.

The results: The authorized users were passive as non-IT testers until it was proven that their work was being registered. There was no planned or overall testing.

The testers found several functional problems, however, and once the system went on line, they helped colleagues to learn how to use it. This improved the speed and quality with which the new system was learned.

Case II

When the software was almost ready, one or two volunteers from each affiliate and department of the company were asked to represent users. They were non-IT testers and were trained on using and testing the system. The non-IT testers had access to the testing environment. First they were allowed to learn about the system on their own, playing around with it and asking the developers for consultations. Then there were planned testing activities. The professional IT testers who were a part of the development group prepared use cases which each non-IT tester had to handle. All of the system's requirements were covered in this way. The non-IT testers were encouraged to supplement the test cases or scenarios from their own experience, choosing test data on the basis of their own views.

Some of the assignments had to be carried out on a specific date and time and by all testers at once. This tested the performance of the system to operate close to true capacity – many connections and many demands simultaneously. User training was also used to test the system's performance.

The volunteer non-IT testers and users helped to test the reaction of the software to major usage – something that is complicated and expensive to simulate.

The results: Later the volunteers helped colleagues to learn how to use the system once it was on line. The carefully considered tests made it possible to examine several aspects of the testing simultaneously. Not all of the non-IT testers operated equally well or with equal motivations, however. The more active volunteers and those who posted the best results in terms of defining problems later took regular part in the testing of new versions of the software.

Case III

Testers at the organization were brought in when the software design was almost completed. They were non-IT testers who had worked as full time testers on the old software. Some of them had 10 and more years of experience with testing, but they all arrived at the process as users, and they had never been properly trained to conduct the tests.

The testers were free to read the recorded system requirements, which were in slightly formalized language. They drew up testing plans, test designs and scenarios. The testers were subordinate to the director of the software development group. Work had to be done quite quickly. There was much personnel turnover in the development group.

The software had few resources to make the work of the testers easier – for instance records in the LOG file or the ability to receive intermediate results of calculations.

There was serious training in the field of testing – more than 40 hours in all. That improved the qualifications of the testers, and it enhanced understanding among testers, developers and bosses, but the change in working culture and processes occurred slowly, and there were no rapid improvements in the quality of the testing.

The results: Because of a lack of time, developers could not support the testers sufficiently, and the testers mostly focused on the most important aspects of functionality. The testers found it difficult to establish test data without a user interface to enter them. Testers also had problems in testing the external interfaces of the system. Serious work began to restructure the whole testing process.

Conclusions

Companies use non-IT testers for various reasons – there are no professional IT testers, it's hard to find them, they cost a lot of money, they don't have a sufficient understanding of the company's business, etc. On the other hand, non-IT testers know a lot about the company's business and can learn and make effective use of many things during the testing process.

Non-IT testers can have problems with documentation, description of test designs, and justifying the test examples which have been chosen. This is largely because testers fear demonstrating their lack of knowledge or skills, and they do not want anyone to look at their mistakes. The company suffers as a result – testers cannot learn from one another. If a tester departs, colleagues have problems in adapting his or her test system. Even the testers themselves cannot always follow along with the systematic aspects of their testing so that the system can be updated when there are changes in the software.

Non-IT testers can best be used in combination with professional IT testers, because they supplement one another. Non-IT users can test the system's correspondence to business processes, the use of the software, and the user interfaces – screen forms and reports.

The work of non-IT testers should be supported by software developers. They can help in placing test data in the database, establishing SQL queries, and handling other issues which require specific IT skills.

Non-IT testers can also establish usage scenarios and test data which are close to reality, taking into account boundary value analysis and analysis of the equivalence classes. Developers often cannot do this, because they know less about the nuances of the relevant area of business. They can prepare simple SQL queries in the database to check the data and to study records in LOG tables.

Non-IT testers can provide good assessment of the applicability of the system's user interface – are work place comfortable, can all functions be carried out, does the system unnecessarily overburden vision or the memory, and is the system easy to use?

Training of non-IT testers in the area of testing does not always improve their work results significantly. When time is short, they cannot make adequate use of their knowledge, for instance.

Non-IT tester training cannot be generalized. It must be adapted to each organization, making use of the software that is being tested. Lots of examples are

necessary. Bosses should take part in the training, because it often lays bare problems about which the bosses had no idea before.

References

1. Black, R.: Critical Testing Processes. Addison-Wesley. 2004
2. Perry, W. E.: Effective Methods for Software Testing. 2nd-ed. John Wiley & Sons. 2000.
3. van Veenendaal, E.: The Testing Practitioner. UTN Publishers, Den Bosch. 2002
4. Kaner, C., Bach, J., Pettichord, B.: Lessons Learned in Software Testing: A Context Driven Approach. Wiley Computer Publishing. 2002.
5. Kaner, C., Falk, J.., Nquyen, H. Q.: Testing Computer Software. 2nd-ed. John Wiley & Sons. 1999.