Collaborative Environment and Agile Development

Bogdan GHILIC-MICU, Marian STOICA, Marinela MIRCEA
The Bucharest University of Economic Studies
ghilic@ase.ro, marians@ase.ro, mmircea@ase.ro

Over time, information and communications technology development has made a direct impact on human activity in the individual context as well as familial, economic and social. This has laid the premise for adoption of new and modern paradigms in individual and organizational activity management. The evolutionary climax of the social universe is called nowadays knowledge society. The knowledge society succeeds the information society, emphasizing the development of the opportunities brought by collaborative work environment and agile approach. In this paper we will highlight the use of collaborative environment in agile software development, as an instrument for managing organizations in knowledge society. Thus, we will emphasize the paradigms of agile testing, validation and verification in collaborative environment.

Keywords: *Collaborative Environment, Knowledge Society, Agile Development, Testing, Validation, Verification, Management, Information and Communication Technology*

1 General Context

From historic perspective, the human society has evolved from agrarian society to craftsman, industrial, capitalist and information society, reaching today the stage of knowledge society. If we were to say what comes next, it would probably be a society of artificial intelligence, limited natural resources consumption or a robot society. We cannot know for sure, but theoretical speculations have been proposed. On the other hand, we can invoke the current context of the knowledge society as a foundation of organizational development on three pillars: collaborative environment, agile approach and modern management based on information and communications technology (ICT).

Unlike the "old" society, today we work and live in a particular technological context, characterized by the global internet network as support for any kind of activity. We can see changes in the nature of work, from traditional office work to telework in virtual offices or telecommuting. These aspects are premises for increasing the exploitation of individual skills and abilities in organizational context and transferring them with the purpose of achieving the business goals. The modern business moves to the sphere of electronic business, strongly relying on technology and internet.

Still, the management activity or the management as an activity does not change in the same radical way. Traditional concepts, methods and techniques are re-wrapped, terminologically beautified and launched on the market in a process specific to marketing. Thus, while Phillip Kotler speaks about marketing management in information society [9] (as the Bible of business men), why can't we speak about management marketing in knowledge society? The so called modern paradigms of contemporary management can be identified terminologically in expressions like fractal model, anthropocentric model, holonic model, cloud model, chaotic model etc. Let's not forget that the term fractal was introduced by Benoît Mandelbrot in 1975, the term holon (as organizational structure) was introduced by Arthur Koestler in 1967 and so on.

What is new in modern management is the way these fundamental concepts from various sciences (mathematics, philosophy, psychology, structuralism, semiotics, cybernetics etc.) are adopted and adapted the management practice. In this sense of exposure, characterized by ICT and internet and modern management paradigms we

DOI: 10.12948/issn14531305/18.2.2014.04

invoke the general context of collaborative environment and agile software development (see also the collaborative paradigm of intelligence). The link between the two subjects of current cognitive development is the fact that both collaborative environment and agile development involve the shared participation of several factors of human nature as well as material, financial and temporal. Economic organizations in current society market must be able to adapt quickly to changes on the market. They must have the ability to exploit any new opportunity in their field or related fields. This ability measures the degree of agility of an organization. An agile business is simultaneously flexible, able to adapt to different economic conditions. The agility of a business is largely determined by the agility of the enterprise architecture employed. In this context, the enterprise architecture is a rigorous description of the internal structure of an organization, its decomposition in subsystems, relations between the subsystems, relations with the external environment as well as principles that must be observed for organization design and evolution. (Figure 1).

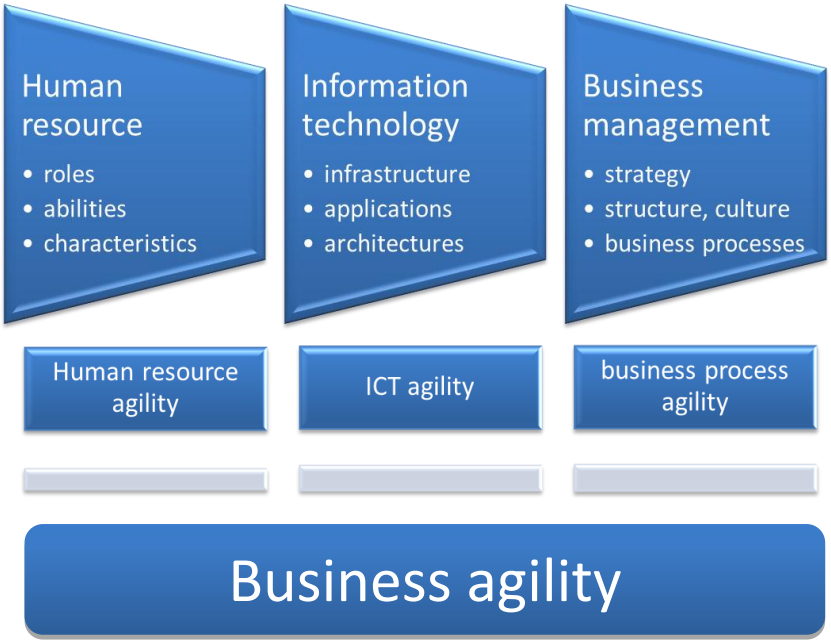

Fig. 1. Business agility in knowledge society

The architecture is built on the terms and specifics of the business's information system. A robust and adaptive information system involves an agile, flexible and adaptive informatics component.

Software applications are the foundation of informatics components. Analysis, design and building these applications takes places more and more in a collaborative environment, relying on agile methodologies (Scrum, XP, LSD etc. – see Figure 2). The quality of agile software developments is ensured by specific testing, validation and verification activities.

DOI: 10.12948/issn14531305/18.2.2014.04

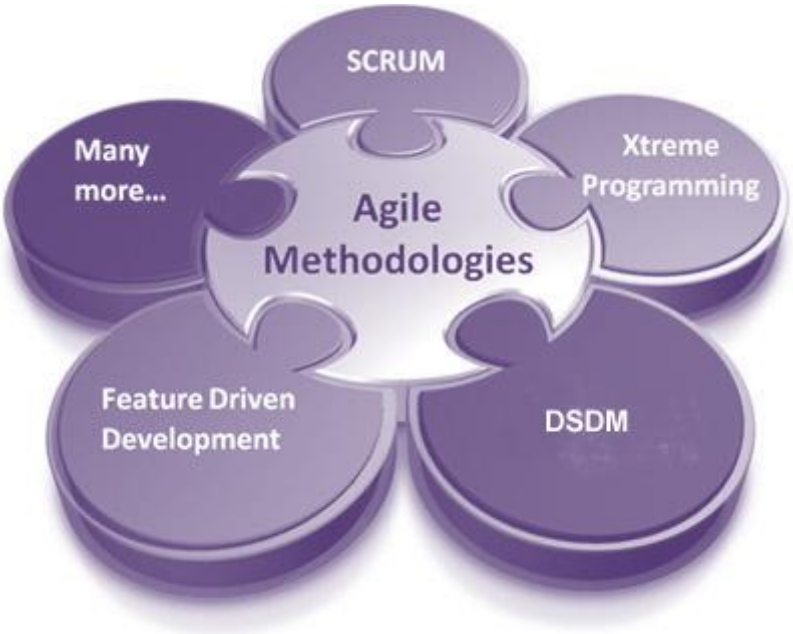

Fig. 2. Agile software development methodologies

2 Why software testing and validation?

Big software system development is a complex and error prone process. Faults may appear in any development stage, and them must be identified and corrected as soon as possible in order to prevent their propagation and escalation of verification costs. Software quality specialists must be involved in early development stages in order to identify the required quality and estimate the impact on the development process [1]. Any quality informatics system professionally designed and developed must be tested and validated before it goes into production. The client must be certain that the system was developed and integrated according to project specification. Also, the client must be certain that the product works correct and without errors.

According to Edsger Dijkstra, "*Program testing may be used to demonstrate the presence of errors, but never to prove their absence!*". Software testing is a new process, or series of processes, designed to verify the degree in which the source code performs as it was designed to and does nothing that was not intended. The software must be predictable and consistent, without surprises for the users. Full testing is impossible, not only in theory, but also in practice. There are two ways to test software: automated testing and manual testing. Each way has its own advantages and drawbacks. [2]

Manual testing requires advances knowledge from the testing specialist, while automated testing can carry out many tests in a short time. Most programmers favor automated testing, but even so, some manual testing is requires to eliminate some errors. Software industry uses both kind of tests, but most persons prefer automated testing to save time and money.

Testing throughout the life cycle ensures the implementation of fundamental concepts and approaches of software testing. The subject of testing is important for two reasons: first, according to recent US government studies, in the year 2000 59.5 billion USD were lost due to low quality software; second, 22.2 billion USD annual loss may be eliminated by implementing appropriate testing in all development stages [3]. Testing is a vital part of software development and must start as soon as possible, becoming an important part of the requirement definition process. In order to achieve the most useful perspective of the development project, the entire life cycle must be designed, including how the user feedback will influence the future demand [4].

Testing is the activity of conceiving test cases, performing the tests and evaluation of

DOI: 10.12948/issn14531305/18.2.2014.04

test results. One test consists of execution of the program for a conveniently chosen input data set in order to verify the result is the expected one. The test case is a set of input data, execution conditions and expected results, designed for a particular purpose like verification of a certain execution path in the program or verification of compliance to a certain condition. [10]

Testing may involve one or more factors, the more the better. Among these factos are [11]:

- ✓ business requirements;
- ✓ functional design requirements;
- ✓ technical design requirements;
- ✓ regulation requirements;
- ✓ source code;
- ✓ business administration restrictions and standards;
- ✓ corporative standards;
- ✓ best practices in the profession of trade;
- ✓ hardware configuration;
- ✓ cultural problems and linguistic differences.

Testing process consists of the following stages (Figure 3):

- ➢ unit testing – testing of individual components;
- ➢ module testing – testing of a collection of related components;
- ➢ subsystem testing – testing the collection of modules integrated into a subsystem;
- ➢ system testing – testing the full system before delivery;
- ➢ acceptance testing – tests performed by actual users to verify if the system complies to the requirements.

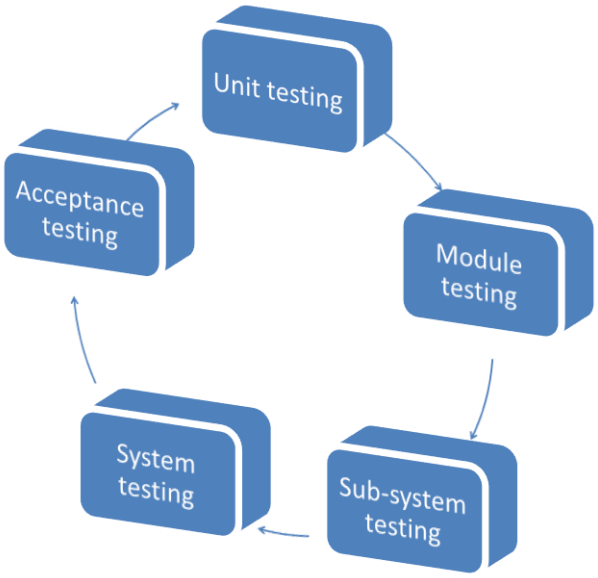

Fig. 3. Test process

The following are the most common *testing principles* [12]:

- ➢ definition of expected results;
- ➢ avoidance of testing own programs;
- ➢ rigorous inspection of each test results;
- ➢ writing the test cases for both valid and invalid input conditions;
- ➢ program testing – verify if it does what specifications require but also if wht the program does is mentioned in specifications;
- ➢ always save the test cases and reuse them;
- ➢ organization and planning of test process, considering there will be errors;
- ➢ testing is a creativity stimulation activity;
- ➢ a test case must define the desired output or result;
- ➢ testing plan must not assume there will be no errors;

DOI: 10.12948/issn14531305/18.2.2014.04

- the probability of finding an error in a portion of source code is proportional with the number of errors already found.

There are also a series of *testing axioms* [13]:

- it is impossible to completely test a program;
- software testing involves assuming a risk;
- testing cannot prove there are no errors;
- the more errors are found, the more there are to be found;
- testing paradox: the more a program is tested, the more its immunity to testing increases;
- not any identified error will be removed;
- specifications for the software product change permanently;
- software testers are not the most appreciated members of the development team.

The following are the *priorities and goals of testing*:

- finding the flaws;
- since a program cannot be completely tested,
  - ✓ test first the most important features and abilities;
  - ✓ test the parts where errors are most likely to occur;
  - ✓ test the integrity level of the software;
- the general testing goal and the purpose of each planned test case must be known (the test case design techniques systematically help achieving the goals and covering the concerns);
- test goals are the test requirements (establishes the functional covering, quality, test set, risks, constraints, required standards).

Testing is an important part of maintaining and improving the quality of software. The product quality improves throughout the development cycle by repeating the cycle "test – identify fault – correct". Quality insurance includes all the steps taken to improve the quality of software.

3 Software Verification and Validation

Verification and validation (V&V) must be conducted in every stage of the development process. V&V process has two main goals: uncovering the flaws of the system and evaluation of the system's utility in operational state. Verification checks the system during development for compliance to the standards, specifications and requirements.

This activity answers questions like:

- ✓ Does the system comply to the specifications?
- ✓ Is the product developed correctly (as it should be)?

Validation determines if the system will be usable on the market. This activity answers questions like:

- ✓ Does the product cover operational needs?
- ✓ The product can be used in the initially established environment?
- ✓ Is the right product being developed?

Verification and validation aim to determine if the system complies to the specifications and meets the clients requirements. V&V involves processes of verification/control, revision and testing the system. System testing involves running it using test cases derived from real data that must be processed.

The distinction between validation and verification is shown in figure 4 and was well defined by Barry Boehm (Professor Emeritus at University of Southern California), who described validation as "*building the right system*" and verification as "*building the system right*". Validation and verification activities complement each other. Verification may involve all stages of the development process, with evaluations of user requirements and design specifications, but users' evaluation is limited by their ability to understand the design and development details. Validation focuses on the final product, which can be extensively tested by users during acceptance test. Users' needs and delays in validation lead to high risks and costs. This is why validation must be associated with specific verification

DOI: 10.12948/issn14531305/18.2.2014.04

activities, which may be conducted during early stages [1].

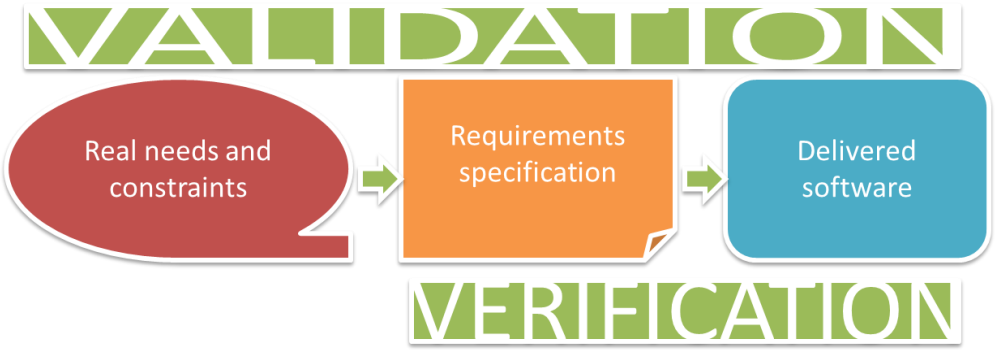

Fig. 4. Verification vs. Validation
(Adapt. from L. Baresi, 96 M. Pezzè/Electronic Notes in Theoretical Computer Science 148 (2006) 89–111)

Verification helps evaluate the product and determines if the product of a certain development stage complies with the requirements even before the stage starts. The activities related to verification of correctness during development are called verification activities. Validation aims to verify if a product meets the client expectations. Validation activities focus on the final product, which is tested from the client's point of view, thus determining if it meets the expectations of global users. Verification may be static or dynamic (Figure 5).

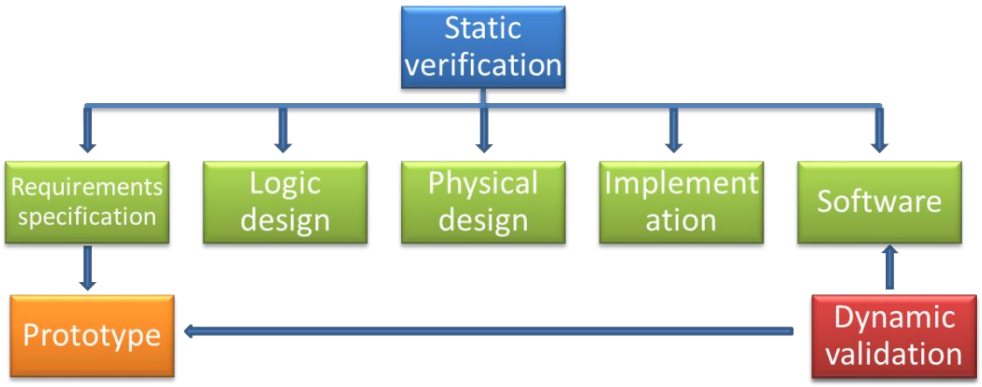

Fig. 5. Static and dynamic V&V

The goal of verification and validation is to decrease the number of errors in the software to an acceptable level. The errors may be present in all stages of the software development cycle.
The usual sources of errors are:

- ✓ errors caused by faulty specifications (most errors);
- ✓ errors caused by design flaws;
- ✓ direct programming errors (relatively few, under 15%).

Validation activities are divided in a) low level testing (test the unit/module, test integration) and b) high level testing (test usability, function, the system, acceptance test).
Testing activity is conducted according to a must-have plan, called ***test plan***. "American National Standards Institute and Institute for Electrical and Electronic Engineers Standard 829/1983 for Software Test Documentation" identify the components describes in table 1, which must be included in a software test plan. The test plan is a final document which describes: the goal, objectives test approach; human resource and equipment engaged in testing; instruments to be used; dependencies and risks; flaw categories; test inputs and output criteria; measurements to be made;

DOI: 10.12948/issn14531305/18.2.2014.04

communication and reports; plan and critical moments.

Table 1. Components included in a test plan*****

Component	Description	Goal
Responsibilities	Specification of persons and their attributions	Assign responsibilities and ensure tracing and concentration
Hypotheses	Code, system state and availability	Avoid misunderstanding s about the program
Test	Test the goal, the plan, duration and prioritization	Sketch the entire process and map specific tests
Communication	Communication plan – who, what, when, how	Everybody knows what to know and when to know
Risk analyses	Critical components that must be tested	Focus on the areas identified as critical for success
Flaw reporting	How the errors will be recorded and reported	Define how to document an error so it can be reproduced, corrected and retested
Environment	Technical environment, data, work environment and interfaces used for testing	Lower or eliminate misunderstandings and potential slowing factors

***** *(Source: http://www2.sas.com/proceedings/sugi30/141-30.pdf)*

4 Software Testing in Agile Development

Software testing in agile development is a complex and controversial problem both in literature and practice. Various persons have various views regarding testing software that was developed with agile methodologies because most agile methodologies do not concern too much with testing activities. The agile model focuses on close collaboration with the client, short iteration and frequent deliveries. Testing a piece of software developed like this is a challenge. Agile strategies do not include many testing practices, which are normally required for quality software.

Agile testing is a software testing practice that follows the principles of agile software development. Agile development integrates testing into the development process. Thus, testing is part and parcel of software development and actively participates in the process of software coding. Agile testing involves an agile and cross-functional team, actively based on specific expertise from testing. Agile teams use a "*whole-team*" approach for "*bake in quality*" of the software product. This approach allows the team to work fast because tests are conducted in real-time, allowing testers to collaborate with the development team and offering them the possibility to identify any problem and transfer it into executable specifications that guide the coding. Testing and coding are conducted incrementally and iteratively, developing every feature until it provides enough value to launch the product. Agile testing has a series of advantages and drawbacks (Table 2). [5]

In an agile development environment, traditional test strategy documents are less frequent and a key agile principle is to favor working on software and not on documentation. Additionally, traditional documentation does not create a common understanding of the testing strategy, due to the fact that document authors (and probably test authors) are usually the only ones that read that document. Still, testing strategy has an increasing role in quality assurance. Automating tests on all levels (unit, integration, acceptance, performance etc.) allows for rapid verification and implementation of software modifications. This provides significant benefits to the business and becomes a more common practice among new start-up companies. [6]

DOI: 10.12948/issn14531305/18.2.2014.04

Table 2. Agile testing features

Advantages	Drawbacks
• Test requirements are discussed and refined by a team (stand-UPS/Scrums), allowing the combined team to better approach the technical/business aspects of the requirements. This allows for general compliance and prevents misunderstandings. • Agile process often requires input and output criteria for stories/descriptions (things to do in a particular launch/iteration). Agile testing ensures that each requirement is well defined and measurable, allowing to determine whether a requirement is fully implemented or not. • QA (Quality Assurance personnel) takes part in writing the requirements. This ensures that test estimation is not overlooked. • Automated testing allows for implementation of regression. • Quality is the responsibility of the entire combined team, not only of QA. The entire team agrees on test strategies, test cases and flaw correction priority plan.	• Estimations and dimensioning of requirements lead to challenges and dependencies. • Right questions are not asked. It is dangerous for QA not to ask the right questions, especially when the described solution is chosen for implementation. Daily meetings of the tam prevent this problem. • New user descriptions may be added in the current iteration. If QA is not included in adding the new descriptions, appropriate engagements and estimations are not taken into account, leading to non-compliance and breaking the deadlines.

A *testing strategy* is a plan document that provides the general direction for a project testing needs. Development of a testing strategy establishes the direction and answers high level testing questions. The value of the testing strategy does not reside in how it is edited, written or formatted, but in planning a test approach. Testing strategies may vary from informal to extremely formal. In its most simple form, testing strategy is just a strategy. It is a roadmap that provides the general direction for what testing must do and provides details on how to do it by answering to the following questions:

- ✓ *how* – the answer identifies the types of tests required, like automated testing or manual testing;
- ✓ *where* – the answer details current testing environment, including the specific server and may include a diagram of all physical and logical components;
- ✓ *who* – must specify the resources used for testing and other resources that may be needed;
- ✓ *when* – o good test strategy describes the moment of the first internal element that foes into testing and may include a tough schedule for the rest of the project.

These high level questions must be answered at the beginning of the project. Testing strategy may also approach issues related to testing like test instrument acquisition or flaw reporting system. The ***goal*** of the testing strategy is to [7]:

- ✓ achieve consensus of goals and objectives from interested parties (for example: management, developers, testers, clients, users);
- ✓ manage initial expectations;
- ✓ insure the right path is being chosen;
- ✓ identify types of tests to be performed on all testing levels; this happens

DOI: 10.12948/issn14531305/18.2.2014.04

always, either in formal or informal way.

A testing strategy provides a *full perspective* over testing and identifies or refers the following:

- ✓ project plans, risks and requirements;
- ✓ relevant regulations, policies and directives;
- ✓ required processes, standards and templates;
- ✓ support guides;
- ✓ interested parties and their testing goals;
- ✓ test resources and their estimations;
- ✓ test levels and phases;
- ✓ test environment;
- ✓ end criteria for each stage;
- ✓ test documentation and evaluation methods.

Testing strategies are ways to approach the testing process. Various strategies may be implemented in each stage of the testing process. From a *typological* point of view, testing strategies include: incremental testing, top-down testing, bottom-up testing, execution thread testing, back-to-back testing.

5. Conclusions

As stated in a paper we recently published [8], no matter what development model is chosen, this activity involves complex processes that are often prone to errors. This is why, beyond agility or traditionalism an important role in software development goes to *testing* and *validation*. Any good quality informatics system, with professional design and implementation, must be tested and validated before it goes into production. When the business evolves in the context of the knowledge society, the collaborative work environment will make its mark on the results testing, validation and verification models.

Acknowledgement

This paper was co-financed from the European Social Fund, through the Sectoral Operational Programme Human Resources Development 2007-2013, project number POSDRU/ 159/1.5/S/138907 "Excellence in scientific interdisciplinary research, doctoral and postdoctoral, in the economic, social and medical fields - EXCELIS", coordinator The Bucharest University of Economic Studies

References

[1] L. Baresi, "An Introduction to Software Testing," *Electronic Notes in Theoretical Computer Science*, vol. 148, no. 1, pp. 89-111, February 2006.

[2] A. Nawaz and K. M. Malik. *Software testing process in agile development*. Internet: http://btu.se/fou/cuppsats.nsf/all/249945527e869a47c125746c0002f4e1/$file/Software_Testing_Process_in_Agile_Development.pdf, June, 2008 [Feb. 27, 2014].

[3] G. Everett, R. McLeod, *Software Testing Testing Across the Entire Software Development Life Cycle*. New Jersey: John Wiley & Sons, 2007.

[4] MSDN, Testing in the Software Lifecycle. Internet: http://msdn.microsoft.com/en-us/library/jj159342.aspx, [Feb. 27, 2014].

[5] Belatrix Software. Agile Software Testing. Internet: http://www.belatrixsf.com/index.php/whitepaper-agile-software-testing, [Feb. 27, 2014].

[6] A. Smith. Agile Test Strategy Template. Internet: http://ennova.com.au/blog/2011/05/agile-test-strategy, May 20, 2011, [Feb. 27, 2014].

[7] Defining a Software Testing Strategy. Internet: http://www.sstc-online.org/proceedings/2002/SpkrPDFS/WedTracs/p616.pdf, 2002, [Dec. 10,2013].

[8] M. Stoica, M. Mircea and B. Ghilic-Micu, "Software Development: Agile vs. Traditional," *Informatica Economică*, vol. 17, no. 4, pp. 64-76, December 2013.

[9] P. Kotler, *Managementul marketingului*, Ed. Teora, ed. V, 2008, ISBN 1594960259

[10] [IEE90] IEEE. Standard: IEEE std 610. In IEEE Standard Glossary of Software Engineering Terminology, 1990

DOI: 10.12948/issn14531305/18.2.2014.04

[11] http://www2.sas.com/proceedings/sugi30/141-30.pdf
[12] G. Myers. *The Art of Software Testing*, 2nd Edition. John Wiley, 2004
[13] R. Patton. Software Testing. Sams Publishing, 2005. Testare agila: http://btu.se/fou/cuppsats.nsf/all/249945527e869a47c125746c0002f4e1/$file/Software_Testing_Process_in_Agile_Development.pdf

Bogdan GHILIC-MICU received his degree on Informatics in Economy from the Academy of Economic Studies Bucharest in 1984 and his doctoral degree in economics in 1996. Between 1984 and 1990 he worked in Computer Technology Institute from Bucharest as a researcher. Since 1990 he teaches in Academy of Economic Studies from Bucharest, at Informatics and Cybernetics Economy Department. His research activity, started in 1984 includes many themes, like computers programming, software integration and hardware testing. The main domain of his last research activity is the new economy – digital economy in information and knowledge society. Since 1998 he managed over 25 research projects like System methodology of distance learning and permanent education, The change and modernize of the economy and society in Romania, E-Romania – an information society for all, Social and environmental impact of new forms of work and activities in information society.

Marian STOICA received his degree on Informatics in Economy from the Academy of Economic Studies, Bucharest in 1997 and his doctoral degree in economics in 2002. Since 1998 he is teaching in Academy of Economic Studies from Bucharest, at Informatics and Cybernetics Economy Department. His research activity, started in 1996 and includes many themes, focused on management information systems, computer programming and information society. The main domains of research activity are Information Society, E-Activities, Tele-Working, and Computer Science. The finality of research activity still today is represented by over 70 articles published, 20 books and over 30 scientific papers presented at national and international conferences. Since 1998, he is member of the research teams in over 20 research contracts with Romanian National Education Ministry and project manager in 5 national research projects.

Marinela MIRCEA received her degree on Informatics in Economy from the Academy of Economic Studies, Bucharest in 2003 and his doctoral degree in economics in 2009. Since 2003 she is teaching in Academy of Economic Studies from Bucharest, at Informatics and Cybernetics Economy Department. Her work focuses on the programming, information system, business management and Business Intelligence. She published over 30 articles in journals and magazines in computer science, informatics and business management fields, over 30 papers presented at national and international conferences, symposiums and workshops and she was member over 15 research projects. She is the author and coauthor of 10 books. In February 2009, she finished the doctoral stage, and her PhD thesis has the title Business management in digital economy.

DOI: 10.12948/issn14531305/18.2.2014.04