Geambasu, Cristina Venera; Jianu, Iulia; Jianu, Ionel; Gavrila, Alexandru

Article

Influence Factors for the Choice of a Software Development Methodology

Journal of Accounting and Management Information Systems (JAMIS)

Provided in Cooperation with:
The Bucharest University of Economic Studies

Suggested Citation: Geambasu, Cristina Venera; Jianu, Iulia; Jianu, Ionel; Gavrila, Alexandru (2011) : Influence Factors for the Choice of a Software Development Methodology, Journal of Accounting and Management Information Systems (JAMIS), ISSN 2559-6004, Bucharest University of Economic Studies, Bucharest, Vol. 10, Iss. 4, pp. 479-494

Accounting and Management Information Systems
Vol. 10, No. 4, pp. 479–494, 2011

INFLUENCE FACTORS FOR THE CHOICE OF A SOFTWARE DEVELOPMENT METHODOLOGY

Cristina Venera GEAMBAŞU[1]*, Iulia JIANU,*
Ionel JIANU and Alexandru GAVRILĂ
The Bucharest Academy of Economic Studies, Romania

ABSTRACT

The success rate of software development projects can be increased by using a methodology that is adequate for the specific characteristics of those projects. Over time a wide range of software development methodologies has been elaborated, therefore choosing one of them is not an easy task. Our research reviews the main categories of development methodologies and then focuses, for a detailed study, on three of them: Rational Unified Process (RUP), Rapid Application Development (RAD) and Extreme Programming (XP). For each methodology it is presented the structure of software life cycle, there are identified the situations in which the methodology can be used successfully and the situations in which it tends to fail. Based on the literature review of software development methodologies and on a series of surveys, published by different researchers, exploring the state of practices in this field, we have identified a number of factors that influence the decision of choosing the most adequate development methodology for a specific project. The methodologies that are subject of this study are evaluated in relation to these factors to find out which development methodology is the most adequate depending on the level of the factors for a specific project. The results of our research are useful for the developers by helping them to identify what software development methodology can be used with success for a specific project.

Software development methodology, Rational Unified Process, Rapid Application Development, Extreme Programming, choosing the adequate methodology

[1] *Correspondence address*: Faculty of Accounting and Management Information Systems, The Bucharest Academy of Economic Studies, 6, Piata Romana, email: cristina.geambasu1@gmail.com

INTRODUCTION

When starting a project that has as purpose the software development, it is very important to use a methodology that increases its success rate. A report of the Standish Group International (2009) on projects success rates shows that 32% of all projects succeeded (delivered on time, on budget, with required features and functions), 44% were challenged (late, over budget, and/or with less than the required features and functions) and 24% failed (cancelled prior to completion or delivered and never used). The use of an adequate methodology plays an important role in developing software, to assure that it is delivered within schedule, within cost and meets users' requirements.

Developers can choose from a wide range of software development methodologies[i]. The present research reviews the main categories of methodologies: "traditional" and "agile". From these categories we have selected, for our study, two representative methodologies: Rational Unified Process (RUP) – a "traditional" methodology and Extreme Programming (XP) – an "agile" methodology. RUP is one of the leading process frameworks (Barnes, 2007) and is used effectively for thousands of projects (Kroll & Kruchten, 2003). XP is the most famous and widely used among agile software development methodologies (Valkenhoef *et al.*, 2011; Rizwan Jameel Qureshi & Hussain, 2008; Angioni *et al.*, 2006). Along with RUP and XP, we have considered for our study a third methodology - Rapid Application Development (RAD). The reason for this choice is that RAD combines elements from both "traditional" and "agile" methodologies.

The main purpose of the present research is to identify and analyze the key factors that influence the decision of choosing the most adequate software development methodology for a specific project. The methodologies that are subject of this study (RUP, XP and RAD) are analyzed in relation to these key factors. The findings of this analyze provides information regarding which methodology is best to be used depending on the level of each factor for a specific project.

1. LITERATURE REVIEW

The main categories of methodologies elaborated over time are subject of many researches (Boehm & Turner, 2004; Nilsson, 2005; Abrahamsson *et al*., 2002; Cockburn, 2002; Cohena *et al*., 2004; Bhalerao *et al*., 2009).

In a first stage the methodologies were highly structured, a great part of the activities of the development process being planned from project initiation. In the scientific literature these methodologies are referred to as "traditional" or "heavyweight". These methodologies require a clearly defined process for developing systems, based on a comprehensive documentation, in order to make this activity more predictable and efficient. A large part of the software process is planned in detail for a long period of

time. The “traditional” methodologies can be used with success only for developing systems for which the requirements are clearly defined from the beginning of the project.

In response to this type of development, have been elaborated methodologies which lead to obtaining software in a shorter period of time, using fewer resources (human, financial, etc.). Known first as "light" methodologies, a series of methodologies are referred to as “agile” after the “Agile Manifesto” reunion in 2001. Abrahamsson *et al*. (2002) believes that a methodology can be considered as “agile” when software development is “incremental (small software releases, with rapid cycles), cooperative (customer and developers working constantly together with close communication), straightforward (the method itself is easy to learn and to modify, well documented), and adaptive (able to make last moment changes).

As regards the three methodologies (RUP, RAD and XP) chosen for our study, a review of the extant literature reveals that most of the works contain only a description of these methodologies, without analyzing the factors that influence the selection of the most adequate one (see Jacobson *et al*.,1999; Kruchten, 2000; Hanssen *et al*., 2005; Barnes, 2007; Morley *et al*., 2002; Avison & Fitzgerald, 2006; Beck, 2000; Beck & Fowler, 2001; Williamsa, 2010)

A series of researches have as subject the factors that influence the selection of the most adequate software development methodology. Russo (1995) concluded, based on a survey of over one hundred organizations, that “the three most important features both for selecting and using the methodologies were: structured development techniques, well-defined corporate policies/procedures, and sharing of information between developers”. Cockburn (2000) identifies two factors that affect what methodology is appropriate: the project priorities and the methodology designer’s peculiarities. However, these researches do not analyze specific development methodologies in relation to these factors to identify which methodology should be used according to the level of the factors.

2. RESEARCH METHODOLOGY

The first part of our research is an overview of the main categories of software development methodologies. Then we have selected three representative development methodologies for our study. These methodologies are analyzed and presented in more detail, outlining their strengths and weaknesses. The literature review of software development methodologies, along with the analysis of a series of surveys, published by different researchers, exploring the state of practices in this field, provided us the necessary information to identify the key factors that influence the decision of choosing the most adequate development methodology for a specific project. In relation to each factor we have evaluated the methodologies that are subject of this research to find out the suitability of a given methodology depending on the level of the factor.

3. SOFTWARE DEVELOPMENT METHODOLOGIES

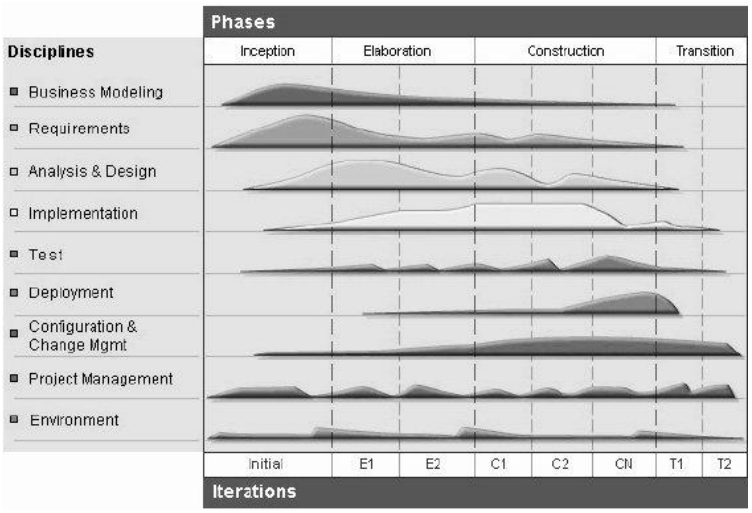
A software development methodology "is a collection of procedures, techniques, tools and documentation aids which will help the systems developers in their efforts to implement a new information system" (Avison & Fitzgerald, 2006).

Our research focuses on three development methodologies: Rational Unified Process (RUP), Extreme Programming (XP) and Rapid Application Development (RAD). Further on, we will synthetize the main characteristics of these methodologies, the way the software life cycle is structured in each of them, as well as their strengths and weaknesses.

3.1. Rational Unified Process (RUP)

RUP provides a framework for the development of information systems, providing a detailed description of the activities to be conducted by the developers. The life cycle of information systems is structured in four phases (Figure 1): inception, elaboration, construction, transition. Each phase if composed of one or more iterations which cover a series of disciplines. A discipline is "a collection of activities that are related to a major area of interest" (IBM Corp., 2006). Each activity is performed by one or more participants that play a certain role within the project and produce or modify one or more artifacts. An artifact is any deliverable result that is used, produced or modified during the software life cycle, such as a report, a document, a use case diagram, a list of risks etc. The disciplines performed within RUP methodology for software development are: business modeling, requirements, analysis and design, implementation, test, deployment, configuration and change management, project management and environment.

Figure 1. **RUP phases and disciplines**



(Source: IBM Corp., 2006: 4)

RUP methodology provides a comprehensive framework for software development that must be adapted according to several factors, such as (Jacobson *et al*., 1999): the size of the operating system, the area in which the software will operate, complexity, experience and skills development team, the way the project is organized.

The process of adapting RUP methodology to fit the specific requirements of a particular project is complex. Hanssen *et al*. (2005) have identified three possible approaches for adapting RUP methodology: adaptation in a single step for each project, defining a subset of the framework adapted to the organization or adaptation by categories of projects.

Villiers (2003) considers that the use of RUP methodology contributes to project success, because it is based on some of the most modern software engineering practices such as: iterative development, requirements management, visual modeling, components based architecture, continuous verification of quality and change control. An important advantage of using RUP is that it imposes risk identification and establishment of mitigation strategies at an early stage, which helps to a more realistic estimation of costs and development time of project. RUP emphasis on accurate documentation and provides a detailed description of activities to be performed, roles related to each activity and artifacts to be obtained.

Beside the advantages, the use of RUP methodology also has some disadvantages. The high complexity of the methodology requires the use of a large number of resources (human, financial, etc.) which makes it difficult to learn and manage. The process of tailoring the methodology is a difficult one, which must take into account many factors in order to avoid the appearance of inconsistencies due to reduction of activities.

3.2. Rapid Application Development

RAD methodology allows rapid development of information systems from the design phase to completion, under conditions of relatively low costs. The software is divided into smaller components, which facilitates making changes throughout the development process. For project components there are defined delivery deadlines (time-boxes) that should not be exceeded. The features are prioritized and requirements are reduced to fit the time if necessary.

RAD methodology uses other methods of the same nature, such as JAD, spiral development process or prototyping. In RAD projects the screens displayed during prototyping become screens of the software. Also, as in other methodologies, it is possible to reuse components.

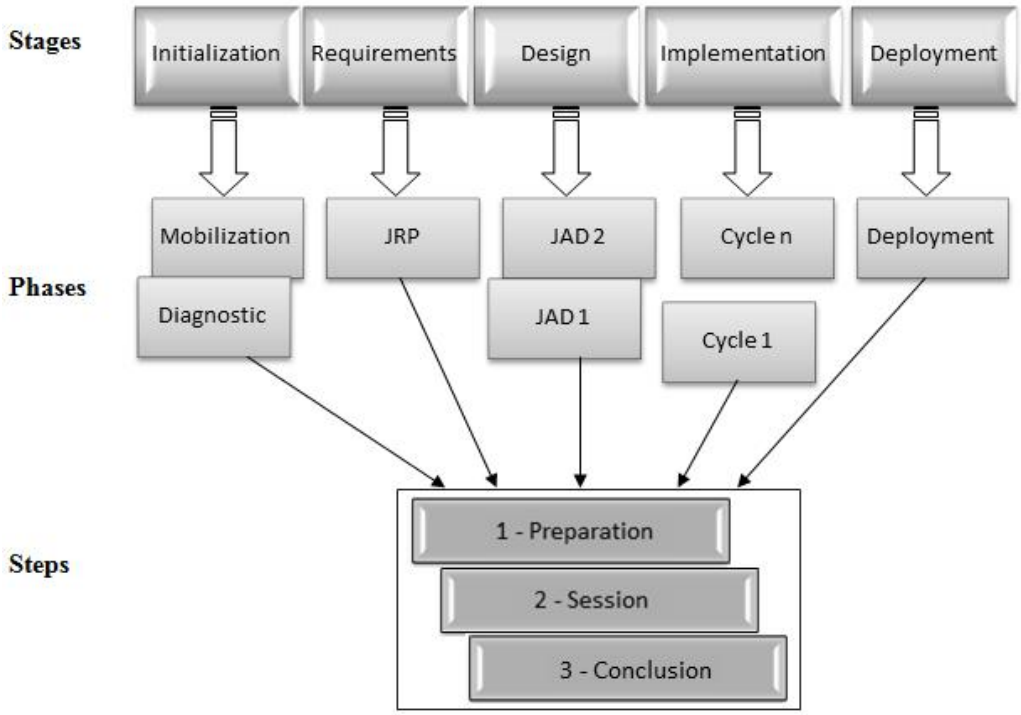
Unlike structured methodologies, which include covering of a great number of steps to obtain a software product, RAD methodology proposes a few steps that actively involve the development team and users, leading to more quickly obtaining of

software. The software life cycle of RAD is structured in five stages (Figure 2): initialization, requirements, design, implementation and deployment.

Each stage involves the execution of one or more phases. Each phase is divided into three steps:

- *Preparation*. Are organized a series of materials to be presented, discussed and modified during the session.
- *Session*. Various participants in the process of software development meet to make decisions on the way the future activities should be conducted.
- *Conclusion*. The session result is formulated under the form of conclusions that will be considered in the process of software development.

Figure 2. **RAD life cycle**



(Source: Morley *et al*, 2002: 131)

In the projects developed using RAD methodology, the responsibilities are clearly allocated among the various participants. So, each participant may play one of the five roles considered by this methodology: binomial project manager (a user project manager and a software engineering manager), user, RAD expert, prototypes developer or owner. RAD methodology concentrates on the user, which is actively involved through the development process, and therefore the user satisfaction is high.

RAD methodology brings many advantages compared to the methodologies used until its appearance, leading to decreasing the time necessary to obtain the final system, costs reduction and mitigation of the risk of failure by including the users in the development team. Due to the use of prototypes RAD allows users to interact with variants of the system from early stages of the development process. The changing requirements can be quickly incorporated in the system.

However, there are several risks to implementing RAD methodology, especially related to the requirements, because usually they are not considered systematically and, as the team is working quickly through project iterations, it is possible to miss significant requirements. The methodology neglects aspects related to systems management (maintenance and reorganization of databases, backing up, restoring after system failures, etc.). The developed system will have less features than in the case of using structured methodologies, because of delivery deadlines (time-boxes).

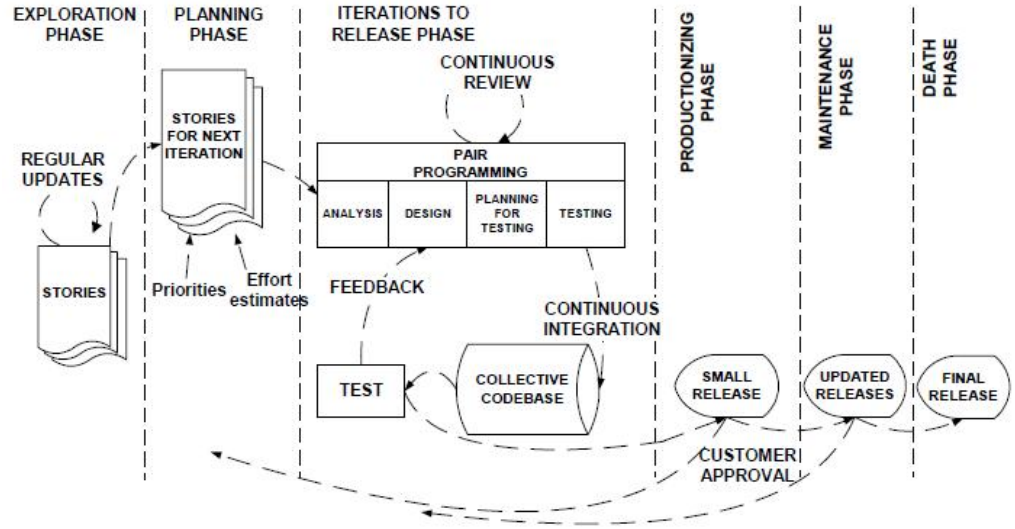
3.3. Extreme Programming (XP)

Beck (2000) defines Extreme Programming as being "a software development discipline that organizes people to create high quality software in a more productive manner". It is considered to be an agile methodology because it is organized into several short development cycles, thus trying to reduce the cost of changes made to adapt to the requirements expressed by customers during the system life cycle. The methodology focuses on the development issues at the expense of the management ones and was designed to be fully or partially adapted within an organization. The working teams are small and are aimed at rapid development of software in an environment where requirements change frequently.

Extreme Programming increases the probability of a long term success of the developed software, by using a set of 12 practices: Planning Game, Small Releases, System Metaphor, Simple Design, Continuous Testing, Refactoring, Pair Programming, Collective Code Ownership, Continuous integration, 40-hour week, On-site customer, Coding standards (Kircher *et al*., 2006). The authors believe that these practices reveal two key assumptions of Extreme Programming methodology: close physical proximity and close customer involvement. The authors state that "a key assumption made by XP is strong and effective communication between the team members, enabling the diffusion of know-how and expertise throughout the group. To enable this strong level of communication among team members, the literature on XP emphasizes that it is important to have the team members physically located close to each other. […] Another important practice of XP requires close customer involvement in the entire development cycle."

The software life cycle of Extreme Programming consists of six phases (Beck, 2000): exploration, planning, iterations to release, production, maintenance and death (Figure 3).

Figure 3. **Life cycle of the XP processes**



(Source: Abrahamsson *et al*., 2002: 19)

Extreme Programming methodology defines seven roles: programmer, customer, tester, tracker, coach, consultant and manager. The team member playing a specific role should have the necessary qualities and characteristics to fulfill the responsibilities related to that role.

Communication, simplicity, feedback, courage and respect are the five values promoted by XP methodology. In the formal methodologies obtaining the software requirements is achieved through documentation. This way of working is different for XP methodology that promotes communication between the customer and the developers as a way of obtaining the requirements. Extreme Programming encourages the use of simple solutions that only implement the user requirements, without adding other features that might be considered useful by the team of programmers. Thus, additional functionalities will be added later as they are required by the customer. Receiving a feedback to confirm that what was done is correct and complete is essential to the XP methodology. The development team receives feedback from both the customer after running the "functional" tests defined by him, and the system by running the "unit" tests that confirm the proper functioning of the software after implementing changes. XP methodology promotes courage in many directions, such as design and implementation of requirements defined on the short term perspective and not on long term, removal of complex code and rewriting it in a simpler way, regardless of the effort required to generate it. Among team members there should be respect, so that everyone can feel valued and motivated to make the effort needed to achieve project objectives.

XP methodology can be used with success only when several conditions are met: teams should have a limited number of members; the physical environment should allow continuous communication and coordination between them; methodology practices and principles must be accepted by all persons involved.

4. FACTORS THAT INFLUENCE THE DECISION OF CHOOSING A SOFTWARE DEVELOPMENT METHODOLOGY

The suitability of a development methodology for a given project is influenced by a series of factors. Based on our researches, we concluded that, of these factors, the most important are: clarity of the initial requirements, accurate initial estimation of costs and development time, incorporation of requirements changes during the development process, obtaining functional versions of the system during the development process, software criticality, development costs, length of the delivery time of the final system, system complexity, communication between customers and developers, size of the development team. Further on, the methodologies subject of our research (RUP, RAD and XP) are analysed relative to these factors.

F1: Clarity of the initial requirements

- ***RUP***. Correct and complete definition of the requirements from the beginning of the project represents the starting point for the development of software that incorporates all the functionality required by the client.
- ***RAD***. This methodology is composed of a variable number of prototyping cycles and consists in building, step by step, viable software. There are held several iterative sessions. The full functional requirements are not set at the beginning of the project, but are specified in detail by the users in each iterative session.
- ***XP***. In the *Exploration phase* of XP methodology the development team members meet with the clients at a planning meeting. The clients define the requirements of the software as "user-stories". It is not necessary that the initial requirements fully describe the functionalities of the final system because the methodology is composed of multiple short development cycles and the requirements are updated in each development cycle.

F2: Accurate initial estimation of costs and development time

- ***RUP***. In the first phase of RUP methodology – *Inception*, is defined the project scope, are identified the risks, is chosen a strategy to mitigate the identified risks, is drawn up an initial model use cases based on the defined requirements and are planned the activities that will be performed during the whole development process. All these elements lead to a realistic initial estimation of costs and development time of project.
- ***RAD***. For each of the five phases components of RAD methodology is established a maximum number of days for accomplishing the objectives of the phase. Depending on the specifics of each project, the effective development time can be

estimated with a small margin of error. The initially estimated development costs are subject to change depending on the effort involved by the implementation of the requirements that are changed during the development process.

- ***XP***. It is very difficult to estimate the effort required for the development of the entire system because not all the requirements are known at the beginning of the project.

F3: Incorporation of requirements changes during the development process

- ***RUP***. The subdivision of phases in iteration allows the developers to make the necessary changes to adapt to new requirements during the whole development process, but the cost of change, especially in the late stages of development, is high. The aspects regarding the management of the changes made during the development process are specified in the *Configuration and Change Management discipline* in RUP methodology.
- ***RAD***. The information system is divided into smaller segments, which facilitates making changes along the development process, at any time in the cycle.
- ***XP***. The methodology is flexible and can easily adapt to changes in the requirements. System changes can be made even in late stages of the life cycle for its adaptation to customer requirements.

F4: Obtaining functional versions of the system during the development process

- ***RUP***. Iterative development of a system leads to multiple versions of the system throughout its life cycle, versions that must be carefully managed to avoid the integration, at the end of the development process, of incomplete solutions.
- ***RAD***. The methodology allows users to interact with variants of the system from early stages due to the use of prototypes.
- ***XP***. Working versions of the developed system are frequently obtained. This way of working helps customers understand the progress in the development of the system and allows him to stop the project after a number of completed iteration if he does not have enough available funds.

F5: Software criticality

- ***RUP***. The methodology comprises a discipline – *Project Management* that aims to track the proper running of the development process by managing the risks from the initiation of the project and by adequately planning and monitoring the iterations in order to achieve project objectives. Risks management involves identification of the risks and elaboration of strategies for their mitigation. *Test discipline* has as purpose to ensure the proper functioning of the system. There are used techniques to check and validate the proper implementation of the defined and designed requirements and there are identified the situations that may cause disruptions. The methodology can be used with success for the development of software with a high level of criticality.

- ***RAD***. During the development process there are numerous tests conducted, but as the team is working quickly through project iterations it is possible to miss information and requirements and the system quality may be lower than in the case of using a traditional approach, as RUP.
- ***XP***. Checking the proper functioning of the developed software is achieved by using two types of tests: "unit test" and "functional test". “Unit tests” are written by developers before adding a new functionality to the system and then run continuously. In this way are checked parts of code (classes, methods, etc.). "Functional test" are specified by the customer and usually refers to checking the functioning of the whole system. Although there are run tests for checking the proper functioning of the software, there still is a risk related to quality assurance because XP methodology does not have a structured review process to reduce the deficiencies.

F6: Development costs

- ***RUP***. The high complexity of the methodology requires the use of a large number of resources, including financial.
- ***RAD***. Due to reusability of the prototypes and to the short development time, the methodology can be used in conditions of relatively low costs.
- ***XP***. The methodology is organized in a number of short development cycles, thus trying to reduce the cost of changes made to adapt to the requirements expressed by the customers during the system life cycle. The development team is small which implies low costs of human resources.

F7: Length of the delivery time of the final system

- ***RUP***. Software development using RUP methodology involves conducting a large number of activities to meet project objectives, leading to extended delivery time of the final system. The methodology can be adapted to fit specific requirements of a particular project, but the adaptation process is also complex.
- ***RAD***. RAD methodology focuses on the limitation of time, being defined delivery deadlines (time-boxes) for project components. If there are problems with meeting the deadlines, the focus is on reducing requirements, rather than on increasing deadlines. Therefore, we can say that the objective of RAD methodology is to deliver the minimum set of requirements necessary in the shortest time period.
- ***XP***. The methodology gives importance to customer satisfaction, by delivering software when is needed and not in a distant future, when the requirements might already be changed.

F8: System complexity

- ***RUP***. The decision to use RUP methodology for software development should take into account the technical and project management complexity. It is recommended to use this methodology when the complexity of both factors is above average.

- ***RAD***. The methodology can be used with success in case of small or medium projects where the scope is well defined. RAD tends to fail when used for the development of complex systems or of distributed systems.
- ***XP***. The methodology emphasis communication between client and developers, thus XP is not recommended for large projects because is difficult to maintain the communication with a large group. Also the use of XP is problematic in case of complex projects with many interdependencies. XP should be used when the system complexity is medium or low.

F9: Communication between customers and developers

- ***RUP***. The customers provide to the developers information on the requirements of the future system and give them, when required, a feedback on certain results of the development process, but they are not actively involved through the whole process of software development.
- ***RAD***. There is a direct participation of customers in the process of software development. The customers participate to working sessions, along with the developers, being actively involved in the process of defining requirements, as well as in the process of evaluating and validating the prototypes and the final software.
- ***XP***. One of XP principles refers to On-site customer. This means that a representative of the future users of the system must be available throughout the development process to answer the questions of the development team.

F10: Size of the development team

- ***RUP***. The high complexity of the methodology requires the use of a large number of human resources.
- ***RAD***. This methodology is not recommended in case of large project teams or when there are many people required to make decisions. It works best with small or medium projects.
- ***XP***. The development team should be small, between 2 and 10 people.

The results of analyzing the software development methodologies in relation to the level of the factors are synthetized in Table 1.

Depending on the level of each factor for a particular project, some methodologies are appropriate while others can lead to project failure. For example, as shown in Table 1, if the “clarity of the initial requirements” is medium or low, it is recommended to use RAD or XP. The use of RUP methodology for a project for which the requirements are not clearly defined at the beginning of the project can determine late delivery, cost overruns, or failure to meet customer requirements.

Table 1. **The level of factors for which the software development methodology is appropriate**

Factor	RUP	RAD	XP
F1: Clarity of the initial requirements	↑	→	↓ →
F2: Accurate initial estimation of costs and development time	↑	→	↓
F3: Incorporation of requirements changes during the development process	→	↑	↑
F4: Obtaining functional versions of the system during the development process	→	↑	↑
F5: Software criticality	↑	→	→
F6: Development costs	↑	→	↓
F7: Length of the delivery time of the final system	↑	↓	↓
F8: System complexity	↑	↓ →	↓ →
F9: Communication between customers and developers	↓	→ ↑	↑
F10: Size of the development team	↑	↓ →	↓

Where,

↓ - Low/ Small
→ - Medium
↑ - High/ Large

CONCLUSIONS

Each project has specific characteristics that should be taken into consideration when choosing the methodology that will be used for software development. This paper provides an overview of the development methodologies, focusing on three representative ones: RUP, RAD and XP. These methodologies are evaluated based on a series of factors that, in our opinion, allow organizations to select the software development methodologies that best fit the characteristics of their project. In some situations, based on the evaluation the factors, it can be concluded without doubt that a certain methodology is most appropriate for software development. In other situations, the evaluation of a part of the factors indicates a certain methodology as being appropriate, while the level of another part of the factors leads to the conclusion that another methodology is suitable. In the second case, the solution is to combine parts of compatible methodologies and use them jointly to develop the software. In either case, choosing the appropriate methodologies is important for the project to be released successfully, on time and within budget.

ACKNOWLEDGEMENTS

This work was supported by CNCSIS-UEFISCSU project number PN II-RU 326/2010 "The development and implementation at the level of economic entities from Romania of an evaluation model based on physical capital maintenance concept."

REFERENCES

Abrahamsson, P., Warsta, J., Siponen ,M.T. & Ronkainen ,J. (2003) "New directions on agile methods: a comparative analysis", *Proceedings of the 25th International Conference on Software Engineering (ICSE'03)*: 244-254

Abrahamsson, P., Salo, O. & Ronkainen, J. (2002) "Agile software development methods: Review and analysis", *VTT Publications 478*:1-112.

Alshayeb, M. & Li, W. (2006) "An empirical study of relationships among extreme programming engineering activities", *Information and Software Technology*, no. 48: 1068-1072

Ambler, S. W. (2002 -2009) "Choose the Right Software Method for the Job", available on-line at http://www.agiledata.org/essays/differentStrategies.html

Ambler, S. W. (2008) "Agile Adoption Rate Survey Results", available on-line at http://www.ambysoft.com/surveys/agileFebruary2008.html

Angioni, M., Carboni, D., Pinna, S., Sanna, R., Serr, N. & Soro, A. (2006) "Integrating XP project management in development environments", *Journal of Systems Architecture*, no. 52: 619-626

Avison, D. & Fitzgerald, G. (2006) *Information Systems Development: Methodologies, Techniques & Tools*, 4th Edition, McGraw-Hill Education

Barnes, J. (2007) *Implementing the IBM® Rational Unified Process® and Solutions: A Guide to Improving Your Software Development Capability and Maturity*, IBM Press

Beck, K. (2000) *Extreme Programming Explained: Embrace Change*, Boston: Addison-Wesley

Beck, K. & Fowler, M. (2001) *Planning Extreme Programming*, Addison-Wesley

Bhalerao, S., Puntambekar, D. & Ingle, M. (2009) "Generalizing Agile Software Development Life Cycle", *International Journal on Conputer Science and Engineering*, Vol. I, no. 3: 222-226

Boehm, B.R., & Turner, R. (2004) *Balancing Agility and Discipline: A Guide for the Perplexed*, Addison-Wesley

Chroust, G. (1996) "What is a software process?", *Journal of Systems Architecture*, no. 42: 591-600

Cockburn, A. (2000), "Selecting a project's methodology", *IEEE Software*, Vol. 17, no. 4: 64-71

Cockburn, A. (2002) *Agile Software Development*, Boston: Addison-Wesley

Cohena, D., Lindvalla, M. & Costa, P. (2004) "An Introduction to Agile Methods", *Advances in Computers*, no. 62: 1-66

Cozgarea, G. (2009) *Metodologii orientate pe obiecte utilizate în proiectarea sistemelor informatice*, InfoMega

Dybå, T. & Dingsøyr, T. (2008) "Empirical studies of agile software development: a systematic review", *Information and Software Technology*, no. 50: 833–859

Fojtik, R. (2011) "Extreme Programming in development of specific software", *Procedia Computer Science*, no. 3: 1464-1468

Gasson, S. (1995) „The role of methodologies in it-related organisational change", *Proceedings of BCS Specialist Group on IS Methodologies, 3rd Annual Conference, The Application of Methodologies in Industrial and Business Change, North East*: 1-2

Hanssen, G. K., Westerheim, H. & Bjornson, F.O. (2005) *Tailoring RUP to a Defined Project Type: A Case Study*, Springer-Verlag Berlin Heidelberg

Highsmith, J., Cockburn, A. (2001) "Agile Software Development: The Business of Innovation", Computer, vol. 34, no. 9: 120-127

Hull, M.E.C., Taylor, P.S., Hanna, J.R.P. & Millar, R.J. (2002) "Software development process – an assessment", *Information and Software Technology*, no. 44: 9-10

IBM Corp. (2006) *Essentials of Rational Unified Process v7.0- Student Guide*, IBM Corp

Jacobson, I., Booch, G. & Rumbaugh, J. (1999) *The Unified Software Development Process*, Boston: Addison-Wesley

Kircher, M., Jain, P., Corsaro, A. & Levine, D. (2006) "Distributed eXtreme Programming", available on-line at http://www.agilealliance.org/show/1057

Kroll, P. & Kruchten, P. (2003) *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*, Addison-Wesley

Kruchten P. (2000) *The Rational Unified Process: An Introduction*, Addison-Wesley, Reading, MA.

Leffingwell, D. & Widrig, D. (2003) *Managing Software Requirements: A Unified Approach*, Addison Wesley

Morley,C., Hugues, J. & Seblanc, B. (2002) *UML pour l'analyse d'un système d'information*, Dunod

Nilsson, A. G. (2005) *Information Systems Development*, Springer US

Novac, C. (2004) "Extreme Programming - a Challange for Software Developers", *Revista Informatica Economică*, Vol. 31, no. 3: 80 - 83

Rizwan Jameel Qureshi, M. & Hussain, S.A. (2008) "An adaptive software development process model", *Advances in Engineering Software*, no. 39: 654-658

Russo, N.L. (1995) "The use and adaptation of system development methodologies", *International Resources Management Association International Conference*, Atlanta, Georgia

Shuja, A.K. & Krebs, J. (2008) *IBM Rational Unified Process Reference and Certification Guide: Solution Designer (RUP)*, IBM Press

Smith, J. (2003) "A Comparison of the IBM Rational Unified Process and eXtreme Programming", available on-line at ftp://ftp.software.ibm.com/software/rational/web/whitepapers/2003/TP167.pdf

The Standish Group International (2009) "CHAOS Summary 2009", available on-line at http://www1.standishgroup.com/newsroom/chaos_2009.php

Tolfo, C. & Wazlawick, R.S. (2008) "The influence of organizational culture on the adoption of extreme programming", *The Journal of Systems and Software,* no. 81: 1956

Valkenhoef, G., Tervonen, T., Brock, B. & Postmus, D. (2011) "Quantitative release planning in extreme programming", *Information and Software Technology*, no. 53: 1227-1235

Williamsa, L. (2010) "Agile Software Development Methodologies and Practices", *Advances in Computers*, no. 80: 1-44

Villiers, D. J. (2003), "Using the Zachman Framework to assess the Rational Unified Process", available on-line at http://www.ibm.com/developerworks/rational/library/content/RationalEdge/archives/rup.html

[i] A reviewer of this paper suggested the use of the term "method" instead of "methodology". When we began writing this paper, we had the same dilemma: should we use the term "method" or the term "methodology". The literature review in the field of software development shows that the opinions are divided. Authors like Avison & Fitzgerald (2006), Cockburn (2000), Russo (1995), Williamsa (2010), Gasson (1995) use the term "methodology", while Abrahamsson *et al.* (2003), Abrahamsson *et al*. (2002); Ambler (2002 -2009), Cohena *et al.* (2004) use the term "method".

The topic has also been approached by Gasson (1995), who states that:

> "There has been some debate about whether the term 'methodology', which literally means "the study of methods", can be used to refer to a particular methodological approach to information systems development. Jayaratna (1994) emphasises that a methodology provides an "explicit way of structuring" systems development […].Maddison et al. (1984) define a methodology as "a recommended collection of philosophies, phases, procedures, rules, techniques, tools, documentation, management and training for developers of information systems"[…]"

After listing some definitions given by several authors for the term "methodology", Gasson (1995) concludes that:

> "a methodology is more than just a method (the 'how' of information systems development), or a process-model. A methodology is a holistic approach: it embodies an analytical framework which is conveyed through intersubjective representational practices and operationalized through a 'toolbox' of analytical methods, tools and techniques."

Due to these clarifications, we think that the term "methodology" can be used within the article in the context of software development.