EXTREME PROGRAMMING AND RATIONAL UNIFIED PROCESS – CONTRASTS OR SYNONYMS?

Ionel IACOB, PhD
Faculty of Computer Science for Business Management,
Romanian – American University, Bucharest, Romania

ABSTRACT

The agile movement has received much attention in software engineering recently. Established methodologies try to surf on the wave and present their methodologies a being agile, among those Rational Unified Process
(RUP). In order to evaluate the statements we evaluate the RUP against eXtreme Programming (XP) to find out to what extent they are similar end where they are different. We use a qualitative approach, utilizing a framework for comparison.

RUP is a top-down solution and XP is a bottom-up approach. Which of the two is really best in different situations has to be investigated in new empirical studies.

1. INTRODUCTION

The agile movement has appeared the last years as an alternative direction for software engineering [1]. Among the agile methodologies, eXtreme Programming (XP) is the most well known [2]. In the current agile boom, many established software engineering methodologies try to present themselves as being agile. The Rational Unified
Processes (RUP) [16] is among those, providing "plugins" to RUP for eXtreme Programming1. Thereby they offer a downsized version of RUP, which is stated to be lightweight, agile style.
Both methodologies share some common characteristics; they are iterative, customer-oriented and role-based. RUP is generally not considered agile; rather it is criticized for being too extensive and heavyweight [21].
RUP comprises 80 artifacts while XP only stresses the code. RUP has 40 roles while XP has five. These issues lead us to the main research question in this paper: Do RUP and XP match together? Are they synonyms, or are they contrasts? IBM states in a white paper that they do [11] but what does a less dependent analysis tell? Our approach to investigate the question is framework analysis.

2. BACKGROUND

2.1. Rational Unified Process

Rational Unified Process (RUP) is a development methodology, developed and marketed by Rational Software, by now owned by IBM. The first release came in 1998 and was a result of cooperation between Grady Booch, James Rumbaugh and Ivar Jacobson [12]. RUP is a general methodology that needs tailoring to specific organizations' and projects' needs. The core values of RUP are [16]
• Use case driven design
• Process tailoring
• Tool support
The process is use case driven, and the use cases constitute the basis for other elements in the development
process. The practical work in RUP consists of the following main items:
• Develop software iteratively
• Manage requirements

• Use a component-based architecture
• Model the software visually
•Verify the software quality continuously
• Manage software change
The RUP methodology is presented using four primary
modeling elements:
• Roles – *who* is doing what
• Artifacts – *what* is produced
• Activities – *how* the work is conducted
• Workflows – *when* a task is conducted
To manage a software project, some kind of a project management model is needed, mostly of a stage-gate type
[5]. This is integrated into RUP, in terms of four phases: *inception, elaboration, construction,* and *transition*.

2.2. Extreme programming
Extreme Programming (XP) is a lightweight development methodology, which stresses teamwork, communication, feedback, simplicity and problem solving [2]. XP consists of a set of software development practices, packaged into wholeness by Kent Beck and Ward Cunningham. Its roots are in the object-oriented community, specifically among SmallTalk programmers.
XP is built on four values:
• Communication
• Feedback
• Simplicity
• Courage
Through communication within and outside the project, it is ensured that the right product is developed.
Quick and frequent feedback provides abilities to correct the direction of the project. Simplicity means building the
right product, not a product for possible future needs.
Courage is needed to maintain openness and communication. To implement the values, 12 practices are defined
which are used in an XP project:
• *Planning Game*
Quickly determine the scope of the next release by combining business priorities and technical estimates. As reality overtakes the plan, update the plan.
• *Small Releases*
Put a simple system into production quickly, and then release new versions on a very short cycle.
• *Metaphor*
Guide all development with a simple shared story of how the whole system works.
• *Simple Design*
The system should be designed as simply as possible at any given moment. Extra complexity is removed as soon as it is discovered.
• *Testing*
Programmers continually write unit tests, which must run flawlessly for development to continue. Customers write tests demonstrating that features are finished.

• *Refactoring*
Programmers restructure the system without changing its behavior to remove duplication, improve communication, simplify, or add flexibility.
• *Pair Programming*

All production code, i.e. code that is actually used in the final product, is written with two programmers at one machine.
• *Collective Ownership*
Anyone can change any code anywhere at any time.
• *Continuous Integration*
Integrate and build the system many times a day, every time a task is implemented.
• *40-hour Week*
Work no more than 40 hours a week as a rule. Never work overtime a second week in a row.
• *On-site Customer*
Include a real, live user on the team, available fulltime to answer questions.
• *Coding Standard*
Programmers write all code in accordance with a set of rules emphasizing communication through code.

XP has four basic activities, *coding, testing, listening* and *designing*, which are conducted by five major roles, *programmer, customer, tester, tracker,* and *coach.*
Iteration is a key concept in XP. The time constant in the different iterations range from seconds to months, see

2.3. Framework analysis

Comparing two methodologies requires some form of empirical studies. Using a quantitative approach [22][18] would require the setup of two parallel projects in an experiment, or launching a case study to investigate certain aspects of one or another of the methodologies. As a lower cost alternative a qualitative approach using frameworks [7] is launched to achieve a first indication of similarities and differences between the methodologies.
A framework provides a simple and structured means for comparisons in a qualitative context. The framework consists of a set of general questions, which are extended with domain-specific questions, in an iterative flexible design fashion. Frameworks have been developed for software engineering by Lindland et al [17] and used by Kitchenham et al to evaluate novel tools [15].

3. FRAMEWORK

The framework used is a combination of two established frameworks, Zachman's and Checkland's [7]. Zachman's framework consists of the six categories
what, how, where, who, when and *why*. Checkland's framework is called CATWOE and has six other categories:
• Client: the stakeholder of the activity
• Actor: the person conducting the task
• Transformation: changes taking place
• World view: what is the outside view of the phenomenon
• Owner: the sponsor of the activity
• Environment: the wider context in which the activity takes place.
Zachman's framework is stronger regarding functions and processes (how, when) while Checkland's framework
is richer on the individuals (client, actor, owner).

4. ANALISYS

4.1. What?

I begin with the history of the methodologies, and then move towards the underlying philosophies and the project types, for which the methodologies are suitable. RUP is created by the well-known triple, Jacobson-Booch-Rumbaugh, launched in its first version 1998.

Jacobson began the development of the use-case based approach at Ericsson in the 1980's. RUP has evolved in conjunction with the Unified Modeling Language (UML) [8]. RUP is based on the originators' and others practical experience from software engineering, and has evolved further during the years, as well as the UML language. RUP is designed for large product development projects.
Even though books are published on the methodology, the main distribution channel is though purchasing of licenses
for the tool support for the RUP methodology, offered by Rational Software, which now is owned by IBM.
XP has its origins in practical applications in projects during the 1990's. Beck and Cunningham have packaged their experiences into XP, originally from a project at Chrysler. It is a lightweight method for small to medium sized software development teams. XP is intended to meet the demands of a context with unclear and volatile requirements. There is nothing commercial in the methodology; instead there is a set of people – a community – who evolve and develop tool support (freeware and shareware) to support XP development projects.
The origin of RUP and XP are similar. They are both based on experience from software engineering. Both are evolved during the same decade, although RUP has its roots earlier.
There are two different underlying philosophies behind RUP and XP. RUP takes to a large extent a technical management perspective while XP focuses on the development staff. RUP is designed to support large projects, while XP is originally designed for small to medium sized projects, for which type of projects several experience reports are published, see e.g. [9][13][19]. The distribution of the methodologies is different; RUP is commercial and XP is freeware.

4.2. Why?

We analyze advantages and disadvantages for the two methods from three perspectives, *technical*, *financial* and *social* points of view.

4.2.1. Technical perspective

On the *technical* side, RUP provides the organization a large package of development tools and documents. It is delivered online via the web, and updated in new releases.
It can be tailored and extended to suit the individual organization's needs. One major sales argument for RUP is the integrated tool-suite.
XP on the other hand strives towards simplicity. It is not connected to specific tools but lets the user choose which tools to use. Tools are developed in the XP community, which support specific practices, e.g. *Junit* for the testing practice.
RUP is a large collection of processes, artifacts and roles. This must be scaled down for most projects except for the very largest ones. XP starts in the other direction, with a minimal core of values and practices, which has to be scaled up to fit larger contexts.

4.2.2. Financial perspective

The *financial* issues are different in the distribution and support of the methodologies, since RUP is a commercial product and XP is freeware. The financial power behind RUP is used for marketing giving more visibility to RUP.
Rational Software is owned by IBM, which has good reputation in the software industry. RUP provides continuous updates, which enables the users being up to date regarding development methodologies.
On the other hand, why should one pay for something that can be achieved for free? Effort must be spent on tailoring RUP, why should an organization then pay for it as well? [10] XP offers the freeware solution, which is financially advantageous, but may cause social reactions.

4.2.3. Social perspective

The *social* aspects of RUP and XP are also related to the commercial versus freeware discussion. Larger software development companies are used to buying software licenses, and hence buying licenses for methodology is quite natural. The freeware principle behind XP is met with skepticism. Can something that is for free be good? The situation is very much like the open source situation. Free software is offered from the open source community and software is licensed from commercial companies, e.g. the Linux operating system versus Microsoft Windows.

The choice is of course primarily technical and financial, but there is a significant social aspect. Smaller organizations and technical staff show a tendency to be more in favor of the freeware/open source approach, while large organization and management are in favor of the license approach. The good reputation and financial strength behind RUP are management arguments, while on the technical level, people know that both approaches need tailoring and hard work – hence they choose the method which is cheapest, least complex, and puts the technical work in focus!

4.3. When and Where?

Regarding the *time* dimension, RUP is organized in four sequential phases, *inception, elaboration, construction* and *transition*. These four phases constitute one development cycle, producing one release of the software. Within each phase, there are a number of iterations, and the four phases have their main focus on different activities, although all activities are run in parallel.

The design evolves as the software evolves. The *simplicity* value and the *simple design* practice emphasize that the design shall be as simple as possible for the current needs, not for future possible needs. Design and analysis activities are not concentrated to the beginning of the project, but intertwined with the development in the *planning* activity. Both RUP and XP stress short iterations, although iterations in XP are even shorter than in RUP. In XP, iterations range from seconds in the pair programming activity, via days in the stand-up meetings to months in a release plan. The iterations in RUP are less frequent, in the magnitude of weeks or months. Both methods strive towards short lead-time and efficient use of resources. The XP principle of developing only what is absolutely necessary, indicates that XP will be the most efficient method. On the other hand, only empirical studies will provide sufficient answers to the question.

The *geographic* dimensions are not explicitly addressed in either methodology, but are present implicitly in both. RUP originates from a context of large distributed development projects, and its approach with artefact-based communication is intended to support this kind of geographical situation. The philosophy behind XP is based on direct, oral communication, both internally in the project and externally towards customers, hence requiring a limited geographical distribution. In practice, XP teams must be located in the very same room to gain the most benefits of the methodology. Even being located at different floors in a building has caused communication problems [14].

4.4. How?

This section deals with the technical content of the two methodologies. We analyze the *extent* of the methodologies, the *organization* of the methodologies and the *tools* support. Regarding the organization, we analyze common aspects, and try to find similarities and differences between the two. The analyzed aspects are *flexibility, project drivers, customer relation, releases and technical work.*

4.4.1. Extent

RUP consists of a large collection of documents, role descriptions, activities etc. RUP stresses the need for tailoring to a specific organization, which in most projects equals downsizing of the methodology. RUP is considered and criticized for being “heavy-weight” [21].

XP is very lightweight, both in its presentation and in the practical application. Everything that is provided to start using XP in a project is covered in each of the sequence of books published on the theme [2][3][4][6]. An indication of the difference in extent of the two methods is illustrated in Table 3, where all the roles of an XP project are presented, with their counterparts in RUP, constituting a small subset of the RUP roles. In total, RUP comprises more than 80 major artifacts, 150 activities and 40 roles [16].

4.4.2. Flexibility

Both methodologies stress the word flexibility. In RUP, it means primarily tailoring to different needs in different contexts and its focus on iterations. In XP flexibility means continuous changes based on the feedback loops. The short feedback loops require continuous actions. The 12 practices can be implemented differently in different projects. The values are the stable core of XP, while everything else may change.

4.4.3. Project drivers

RUP is use case driven, i.e. descriptions of use of the system are implemented, and continuously integrated and tested. XP applies test-driven design, i.e. test case are derived and implemented before the code is written. XP has user stories to guide what to implement. These user stories are less extensive descriptions, compared to the RUP use cases, where the complete scenario for the interaction between the user and the system is defined.

Regarding planning, both methodologies agree on that a complete project cannot be planned in detail. RUP proclaims continuous changes in the plans, while XP advocates planning only the very near future in detail.

4.4.4. Customer relation

Regarding the customer relation both methodologies stress the importance of a close relation to the customer, but still this issue is very different.

XP assumes the customer be involved in person in the team to „answer questions, resolve conflicts and set small-scale priorities" [2]. This is later turned into "an XP project is controlled by an assigned person, defining requirements, setting priorities and answering questions from the programmers". RUP is more flexible on the implementation of the customer relation. It is not always possible or even feasible that the customer is present in person.

4.4.5. Releases

RUP defines a release to be "a stable, executable version of a product and its necessary artifacts" [16], while XP defines it to be " a set of user stories creating a business value" [4]. The XP practice *small releases* and the RUP item d*evelop software iteratively* are very similar, assuming that a release can be both internal and external.

4.4.6. Technical work

XP involves two controversial practices, *collective ownership* and *refactoring,* which are tightly connected. They are also highly dependent on the *continuous integration* and *testing* practices, which constitute the quality assurance mechanisms. In RUP, which originates from larger systems, different project members are responsible for different subsystems.

4.4.7. Tools

The RUP process as such is guided by a tool, and there are suitable tools for e.g. modeling that interface smoothly with the methodology. As the methodology is so extensive, this is absolutely necessary, to guide the user. This is also a part of the commercial success of RUP. XP does not proclaim any specific tools. There are tools offered by the community, e.g. *Junit*, but any kind of CASE tools and project management tools can be used in XP. However, it is worth noticing, that in its original form, whiteboards, paper cards and pens are the most mentioned tools in XP.

5. CONCLUSIONS

In this paper, we have analyzed the similarities and differences between RUP and XP methodologies, based on a framework. Although many keywords and key values are the same, the two methodologies are quite different.

Common values are user/customer involvement, iterations, continuous testing and flexibility. The implementation of these values are however very different. RUP offers an extensive process description, comprising artefacts, roles, activities, integrated toolsuites etc. XP on the contrary stresses values and principles, rather than prescriptive instructions, and focuses freedom and simplicity. The distribution channels are different, RUP being a commercial product by a large company, and XP is freeware, maintained by a community of volunteers.
I conclude from this analysis that the two in many aspects are in contrast. The situation is very similar to the Windows vs. Linux case. One is commercial, the other is freeware. One tends to be advocated by managers, the other by engineers. Still both are operating systems for personal computers. It is important to be aware of this social aspect in the selection of RUP or XP. Which of the two is best suited for certain types of projects needs to be further investigated in empirical studies.

REFERENCES

[1] **Agile Manifesto**, http://www.agilemanifesto.org/ last visited
04-08-20

[2] Beck, K., **Extreme Programming Explained – Embrace Change**, Addison-Wesley, 2000.

[3] Beck, K. and Fowler, M., **Planning Extreme Programming**, Addison-Wesley, 2000.

[4] Chromatic, **Extreme Programming Pocket Guide**, O'Reilly, 2003.

[5] Cooper, R. G., **Winning at new products**, Perseus Publishing, 3rd edition 2003.

[6] Crispin, L. and House, T., **Testing Extreme Programming**, Addison-Wesley, 2003.

[7] Feller, J. and Fitzgerald, B., "**A Framework Analysis of the Open Source Software Development Paradigm**", *Proceedings of the 21st International Conference on Information Systems (ICIS)*, ACM, pp. 58-69, 2000.

[8] Fowler, M. and Scott, K., **UML Distilled. A Brief Guide to the Standard Object Modeling Language**, Addison-Wesley, 2nd edition 1999.

[9] Fraser, S., Beck, K., Cunningham, W., Crocker, R., Fowler, M., Rising, L., and Williams, L., "**Hacker or hero? - extreme programming today (panel session)**", Proceedings of the conference on Object-oriented programming, systems, languages, and application (Addendum) pp. 5- 7, 2000.

[10] Henderson-Sellers, B., Due, R., Graham, I. and Collins, G., "**Third Generation OO Processes: A critique of RUP and OPEN from a Project Management Perspective**", Proceedings. Seventh Asia-Pacific Software Engineering Conference, pp. 428-435, 2000.

[11] IBM (Smith, J.), **A Comparison of the IBM Rational Unified Process and eXtreme Programming**,
http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/TP167.pdf

[12] Jacobson, I., Booch G. and Rumbaugh, J, **The Unified Software Development Process**, Addison-Wesley, 1999.

[13] Karlström, D., "**Introducing Extreme Programming - An Experience Report**", Proceedings Third International Conference on eXtreme Programming and Agile Processes in Software Engineering, 2002.

[14] Karlström, D. and Runeson, P., "**Integrating Agile Software Development into Stage-Gate Managed Product Development**", technical report CODEN : LUTEDX (TETS-7203) / 1-34 / (2004) & local 16, 2004.

[15] Kitchenham, B., Linkman, S., and Linkman, S., "**Evaluating Novel Software Engineering Tools**", Proceedings The 7th International Conference on Empirical Assessment in Software Engineering (EASE 2003), Keele University, Staffordshire, UK, pp. 233-247, 2003.

[16] Kruchten, P., **The Rational Unified Process – An Introduction**, Addison-Wesley 2nd edition, 2000.

[17] Lindland O. I., Sindre, G. and Sølvberg, A.,
"**Understanding in Conceptual Modeling**", *IEEE Software*, March, pp. 42-48, 1994.

[18] Robson, C., **Real World Research**, Blackwell Publishers, Oxford, 2nd edition, 2002

[19] Schuh, P., "**Recovery, Redemption, and Extreme Programming**", *IEEE Software* December, pp. 34-41, 2001.

[20] Scott, K., **The Unified Process Explained**, Pearson Education, 2001.
[21] Tallungs, P, “**Se upp med RUP-mupparna!**”, http://computersweden.idg.se/ArticlePages/dynamic.asp?404;http://computersweden.idg.se/ArticlePages/200304/09/20030409165354_CS491/20030409165354_CS491.dbp.asp, last visited 2004-08-20.
[22] Wohlin, C., Runeson, Höst, M., Ohlsson, M. C., Regnell, B. and Wesslén, A., **Experimentation in Software Engineering: An Introduction**, Kluwer Academic Publisher, USA, 2000.