An Approach for Selecting Software Development Methodologies

Ghassen TOUIL
Computer Science Department, Faculty of Science
M'hamed Bouguerra University, Avenue de l'indépendance, 35000, Boumerdes. ALGERIA
touil_ghassen@yahoo.co.uk
Mohamed AHMED-NACER
Computer Science Department, Faculty of Computer Science and Electrical Engineering
Houari Boumediene University, BP 32 El Alia - Bab Ezzouar. Algiers. ALGERIA
anacer@cerist.dz

Abstract - *In recent years, we have witnessed a growth and diversity in software development methodologies. Underlying principles make software development methodologies different and define a range of software projects that can be dealt with. In the present study, we provide some guidelines that assist organizations to make decisions about the methodology to be used for developing a given product. A framework of factors in relation with methodology, project, and organization was provided and applied to compare the following four methodologies: Rational Unified Process, Extreme Programming, Cleanroom Software Engineering and Open Source Development. The Balanced Scorecard model with its four complementary perspectives was used to guide the selection process. The application of such a model was presented and illustrated in a case study for selecting a project methodology.*

Keywords: ***Software Process Comparison, Software Process Selection, Balanced Scorecard.***

1 Introduction

The diversity of application domains of software and the priorities of project managers resulted in different approaches to developing software. These approaches or methodologies require different types of resources and yield different kinds of products. Project managers should reflect on the methodology to be used and decide upon the right or the most suitable one for their projects. Priorities of the project and the organization will greatly influence project managers' choice [1],[2]. Viewing the large number of methodologies available today, many software engineering researchers have been interested in comparing methodologies [3],[4],[5]. By studying and comparing methodologies, we can understand the context in which a methodology might be applied and yield better value. Such a comparison can assist organizations and project managers in choosing the right methodology for every project.

In our research we deal with methodologies from the user's point of view. The user is represented by the project manager and team members, willing to get the most out of the methodology to be used. Four well-known methodologies, which represent the mainstream in software development, were chosen to conduct this study. These are: Rational Unified Process (RUP), Extreme Programming (XP), Cleanroom Software Engineering and Open Source Development (OSD). Our aim is to evaluate different aspects of the four methodologies and to provide a comparative analysis which can serve as a framework for selecting software development methodologies. A new approach for selection based on the Balanced Scorecard model is introduced. This approach provides guidance in selecting methodologies by balancing different types of objectives belonging to both technical and business aspects. Besides, it takes into consideration the future developments and projects of the company and does not focus only on a single project ensuring therefore continuity in achieving the company's objectives.

The structure of this article is as follows. The second section presents briefly the four methodologies to be compared. In section three, factors for comparison, composing a framework, are identified and used to compare the four methodologies. This comparison is followed by a summary of the obtained results. The fourth section introduces the new approach for selecting software development methodologies using the results obtained in section three. A set of recommendations on the use of the proposed approach is provided thereafter. To illustrate how to use this approach, a case study was conducted. This includes a description of a chosen organization and a product to be developed, followed by recommendations on the methodology to be used.

2 The Software Development Methodologies

2.1 Introduction

Four processes were investigated in this article. RUP was chosen because it is used by thousands of companies worldwide. XP is a revolutionary way of developing software. It stands as a rival to RUP and tries to solve some of RUP's drawbacks as stated by its authors. Open Source is an international trend and is totally different from "classical" approaches to developing software and has many special characteristics. Besides, an increasing number of companies are supporting it. Cleanroom Software Engineering is an approach which took its principles from the semiconductor industry and promises the production of high-quality software. These four software processes represent the mainstream in software development processes.

2.2 Rational Unified Process

The Rational Unified Process is a software engineering process. It provides a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software that meets the needs of its end users within a predictable schedule and budget [6]. The RUP guides software practitioners in effectively applying modern software best practices, such as developing iteratively, taking an architecture-centric approach, mitigating risk at every stage in the process, and continuously verifying the quality of the software [7].

2.3 Extreme Programming

Extreme Programming (XP) is one of the best-known "agile" methods. XP is based on four values and an initial set of twelve practices which was proven useful in improving software development. The fundamental XP values are: communication, simplicity, feedback, and courage. The four values lead to a set of 12 practices, which essentially form the rules of XP [8]. The 12 practices are : planning game, small releases, metaphor, simple Design, testing, refactoring, pair programming, collective code ownership, continuous integration, sustainable pace, on-site customer, and coding standards.

2.4 Cleanroom Software Engineering

Cleanroom software engineering is a theory-based, team-oriented process for development and certification of high-reliability software systems under statistical quality control. A principal objective of the Cleanroom process is the development of software that exhibits zero failures in use [9]. The Cleanroom software engineering process combines formal methods of software specification, design, and correctness verification with statistical usage testing for quality certification [10].

2.5 Open Source Development (OSD)

The concept of Open Source software development has recently gained popularity and acceptance within the software engineering community. An open source methodology is being shaped as a formal way of developing software. A description of the development process is given in [11]. The process starts with a team of developers designing and coding the software. They debug the software until the source code is acceptable. The source code is then released to the general user community, who in turn adds more functionality. The original team plays the role of project coordinators and planners.

3 Comparing Methodologies

3.1 Introduction

The current comparison aims at providing a framework of factors that guides organizations in selecting methodologies. Primary factors were chosen in the present study in relation to the project, methodology, and organization:

- **Project factors:** Software quality required, project innovativeness, project domain, and project criticality;
- **Methodology factors:** Tools support, return on investment of software process improvement, software process improvement and capability and tailorability and adoption;
- **Organization factors:** Team size, discipline and available resources.

These factors represent the most important elements that may affect the choice of the project manager. Each factor was assessed independently from the other factors and contrasted in the context of the four methodologies. The selected factors and the comparison are presented in what follows:

3.2 Project

3.2.1 Software Quality Required

RUP - Verifying quality continuously is one of the six best practices adopted by RUP. Quality should be reviewed with respect to requirements based on different dimensions: reliability, functionality, performance and usability, and different types of tests: unit testing, integration testing, system testing and acceptance testing. RUP assists the user in the planning and execution of all these test types by providing suitable techniques and tools.

XP - XP is very strong on producing high-quality software. It focuses on both aspects of quality: meeting customer expectations and defect counts by combining its 12 practices. Quality is improved throughout all the development process by resolving defects early and providing constant feedback on the product.

Cleanroom - Cleanroom focuses on defect prevention rather than defect correction. By using a compromise of formal specification, statistical testing and verification we may reach a zero defect software. [12] provides a list of projects where Cleanroom was applied resulting in a very low testing failure rate.

OSD - There is no system level testing. Quality is achieved by continuous testing and frequent releases. The quality of OS software is reported to have less bugs than products developed in traditional ways. Requirements are subject to change during the development process. Functionality of the final product may be totally different from what was specified.

3.2.2 Innovativeness

RUP - The first step in a new RUP project is conducting an assessment of your project to test its feasibility. The assessment results in an implementation plan which shows all necessary resources and support for accomplishing the project. Regarding legacy system, RUP does not explicitly address how to deal with them. An extension to RUP was provided by Ronin International, Inc [13]. The extension proposed supports maintenance and after-development activities.

XP - On-site customer principle allows XP teams to get a clear idea on the system functionality. The customer functions as part of the project team and provides required details. The refactoring practice demonstrates XP's strength when dealing with existing systems. With skilled developers, refactoring is an excellent tool to improve legacy code.

Cleanroom - Dealing with legacy code and rewriting existing system is discussed in [14]. Rewriting is limited to systems developed using the Cleanroom approach. This includes small modifications, partial rewrites and adding new components.

OSD - Most OS software deal with system software (operating systems, web browsers, compilers, etc.) where developers have enough knowledge about requirements and architecture [15]. With innovative products and a distributed community it is extremely difficult to elicit requirements to be understood by programmers. On the contrary, the modification of software is one of its great advantages over other approaches

3.2.3 Project Domain

RUP - RUP may be used in various domains of application and for large and small projects. It is used in many companies in different domains: Telecommunication, transportation, aerospace and defense, manufacturing and finances.

XP - Although it has been successful in many domains of application, no clear limits have yet been identified to XP applicability.

Cleanroom - Cleanroom is most suitable for critical applications. Critical refers to applications where defects can cause loss of life or critical financial loss. Space and defense, telecommunications, and system application are the main domains where Cleanroom proved successful. Cleanroom is used in commercial companies but on a very limited scale.

OSD - The nature of OSD makes it target large projects with distributed communities. Most known OS projects are focused on large software development tools and Internet based products. OS can fit for specific domains but the main difficulty lies in finding and motivating interested and skillful developers.

3.2.4 Project Criticality

RUP - Primary goals of plan driven methods are predictability, stability, and high assurance. RUP is no exception. With its risk-based approach to software development, thorough documented plans and specifications, and an increasing process capability through standardization, measurement and control, RUP suits better for developing highly-assurance, safety-critical systems. With low criticality products, RUP becomes time-consuming and shows less efficiency due to the extensive documentation that should be generated.

XP - XP is still untested on safety-critical products. The main difficulties rise from the simple design and the lack of documentation [16]. However, with low criticality systems, XP seems to give better results than plan driven methods.

Cleanroom - Cleanroom software engineering was designed to develop software that shows no failures in use. Incremental development under statistical process control with formal specification and verification of software maintain control over projects. This results in reducing risk and achieving productivity in software development and reliability in software performance.

OSD - OS projects are undertaken by volunteers with no project plan, no schedule, no system level design or detailed design, and no list of deliverables [17]. Geographical limitations between developers add more uncertainties to the development process and make it less predictable with less assurance.

3.3 Methodology

3.3.1 Tools Support

RUP - RUP provides a full software support that helps to automate steps in many activities in the development process. These tools are grouped into a suite that is called IBM Solutions. This includes RUP Builder to compose and publish your own configuration of the Rational Unified Process, Rational XDE for visual modeling and Rational ClearCase for configuration management.

XP - It does not provide or recommend any supporting tool. It is up to the user to choose appropriate tools.

Cleanroom - A set of tools called toolSET is used to support specification, development and certification activities. This includes the following tools:

- **Certification Assistant:** Automatically generates test cases from usage probability distribution and carries out statistical analysis of test results.
- **toolSET_Certify:** A CASE tool for Statistical Usage Testing.
- **CleanTest:** Generates statistical test cases based on input profile.

OSD - Different free tools are used such as browsers (Netscape), text editors (Emacs), compilers (GNU Make), configuration management tools (CVS), etc.

3.3.2 Return on Investment of Software Process Improvement (SPI)

RUP - Best practices of software development recommended by RUP are proven practices that support business and technical decisions. On the other hand, using the integrated tools of RUP product (IBM Solutions) can provide a high return on investment by automating tasks.

XP - XP does not usually perform return on investment analysis to determine an optimal allocation of resources to deliver specific value. XP teams "Do The Simplest Thing That Works". They try to avoid costs of not yet needed functionality. Through the focus on simplicity and rapid delivery, XP reduces lead times which lead to lower investment and reduced operating expenses.

Cleanroom - Cleanroom achieves better return on investment by avoiding post-production defect correction (finding and correcting defects, tracking problems and distributing fixes). This reduces rework and results in a sharp reduction in direct costs of defect correction over the market lifetime of products.

OSD - Costs of training to switch to OSD yield a low return on investment for the development of a single application. If we extend the evaluation to the whole life of the application, or more, after developing other applications, the cost of training becomes marginal and we can get benefit from other programmers doing part of the job for free.

3.3.3 Software Process Improvement and Capability

RUP **-** Organizations using RUP may reach level 2 and 3 of SW-CMM by complementing some project management aspects [18]. Little research has been done to assess RUP with CMM levels 4 and 5. Most of the concerns of the SW-CMM at these two levels are related to the organization. Processes should be implemented to satisfy corresponding key process areas.

XP - XP meets most key process areas (KPAs) of level 2 and 3 of the CMM. The missing KPAs (Software Subcontract Management for level 2 and Training Program and Integrated Software Management for level 3) should be addressed using a proper management support. Beyond level 3, XP ignores or partially covers KPA of CMM. Any organization aiming to reach level 4 or 5 of the CMM should use its own resources to satisfy these KPAs. More details are given in [19].

Cleanroom - The SEI has developed a Cleanroom Reference Model (CRM) that provides a framework for developing a project or organization level Cleanroom process. Once tailored, Cleanroom implements a majority of the CMM of software development. Furthermore, Cleanroom addresses processes that do not have matching KPAs in the CMM [20].

OSD - A formal process called Open Source Maturity Model (OSMM) [21] is used to assess the maturity level of OS software but not the process of development. Some work has been made to relate OSD with the Capability Maturity Model. Assessments have been done for organizations having a defined process and willing to migrate to OSD. A case study was presented in [22]. The study concluded that the process capability decreased from level 3 to level two, while it is possible to reach again level 3 when the development team copes with the readjustments of process management.

3.3.4 Tailorability and Adoption

RUP - The Rational Unified Process could be used in whole or in part "out of the box," as stated in [6]. It must be configured and tailored to the specific context and needs of an organization before its full implementation. One of the major drawbacks of RUP is that it gives no guidelines on its implementation. External expertise may be required.

XP - To adopt XP, you should use it. Practices should be adopted gradually. You should adopt one practice at a time, always addressing the most pressing problem for your team. Once finished, you would go on to the next problem. In practice, we can rarely adopt all the practices but we can get instead a partial adoption.

Cleanroom - According to [23], Cleanroom can be adopted in three ways: partial, complete or advanced. The author suggests a phased approach to implementation by introducing first fundamental Cleanroom principles and key technologies. As team experience and confidence grow, increased precision and rigor can be achieved in a full implementation of Cleanroom technology. Finally, an advanced implementation can be introduced to optimize the Cleanroom process.

OSD - Adopting OSD means opening the source code of your software under a certain license. The main difficulty in adopting Open Source lies in finding and motivating contributors and ensuring its continuity.

3.4 Organisation

3.4.1 Team size

RUP - Two or more team members. There is no upper limit for RUP teams. It scales better to large projects; the larger the team, the more efficient the process is.

XP - XP is aimed for small and medium sized teams. An ideal XP team should be limited between three and twenty project members. Larger teams are possible but tend to fail applying XP practices and principles.

Cleanroom - Small teams between six and ten members. With larger projects, teams may be subdivided into smaller ones.

OSD - Programmers freely contribute to OS projects. This constitutes distributed teams of volunteers. The number of members ranges from few to many thousands.

3.4.2 Discipline

RUP - RUP imposes discipline as a factor for the success of any project. Each member does a well-defined task with a limited knowledge of what others are actually doing until the process becomes standardized. After standardizing processes, change becomes difficult and time consuming.

XP - It is most applicable to turbulent, high change environments [8]. It gives better results with people willing to collaborate. It is an "antidote to bureaucracy" of plan driven methods. People feel comfortable and empowered by having many degrees of freedom.

Cleanroom - Cleanroom views software development as an engineering discipline and not as an art or craft. This gives little freedom to creative work. The development process is achieved in a very disciplined way.

OSD - The system is built by a large number of volunteers. Work is not assigned. Instead, people themselves choose the task they are interested in.

3.4.3 Available Resources

RUP - RUP is a proprietary product. Adopting RUP requires investing a lot of resources. This includes buying the process description, tools support and costs for training people to use these tools. Additional costs may be required to tailor RUP to fit the organization's needs. The process of tailoring RUP is a project itself and may require external expertise.

XP - Training people and consulting are the only requirements.

Cleanroom - Tools support should be acquired. Training is required for learning specification and verification techniques.

OSD - No costs are imposed. Some OS projects may be funded by organizations to encourage people to participate. Money may be spent to hire programmers for instance.

3.5 Summary

A summary of the previous comparison is presented here:

- **Quality:** Cleanroom suits better projects requiring a very high level of quality. RUP and XP can also ensure a high level of quality. OSD can provide a code with a low rate of bugs but suffers from poor requirement elicitation.
- **Innovativeness:** RUP and XP are better at dealing with legacy systems. OSD is good at modifying old code but the absence of documentation limits this advantage. Cleanroom can only deal with systems using the same techniques of specification and certification.
- **Domain of application:** RUP and XP have a wider range of application than OSD which is used mainly for Web products and operating systems. Cleanroom targets a very small range of applications such as real-time systems.
- **Criticality:** RUP and Cleanroom fit better with critical projects. XP and OSD fit better low criticality projects and yield better results than plan driven methodologies.
- **Communication and discipline:** XP and OSD rely more on human factors. Free communication and staff motivation are important to the success of the project. RUP and XP rely instead on discipline and good planning.
- **Team size:** RUP and OSD work well with larger teams. XP and Cleanroom usually target smaller ones.
- **Available resources:** RUP and Cleanroom with less degree require large investment especially for training and tool support. XP and OSD on the contrary do not require much investment.
- **Tool Support:** RUP, Cleanroom, and OS provide necessary tools support to assist the development activities. XP relies more on the member's skills and does not provide such a support.
- **Return on investment:** OSD provides higher long-term return on investment. RUP, XP, and Cleanroom do provide more or less high return on investment.
- **Software process improvement:** Cleanroom's straightforward application leads to a very high level of maturity. RUP and XP can reach high levels of maturity with special arrangements. OSD does not focus on process maturity and does not promote higher levels of maturity.
- **Tailorability and adoption:** RUP shows more flexibility in adoption. Cleanroom provides sufficient guidance on its adoption but in practice it needs a lot of effort to apply its techniques and practices. OSD and XP do not require special organizational arrangements to use them. Their adoption is a matter of willingness and acceptance within the company.

4 Selecting Methodologies

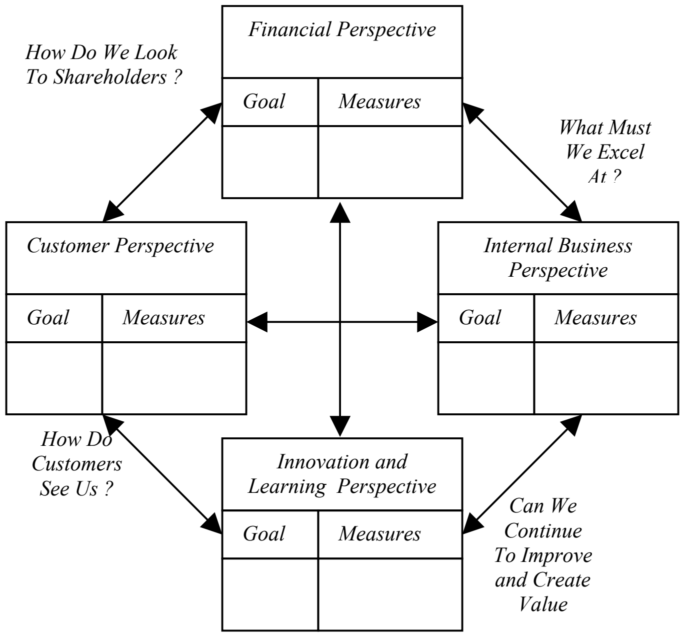
4.1 Introduction

In this section we present a new approach for selecting software development methodologies. The presented approach is based on the Balanced Scorecard model (BSC). This latter combines aspects of software development and business. In the first section we introduce the Balanced Scorecard model and show how to use it for evaluating and comparing methodologies. The second section presents a case study made at a software development company to illustrate the use of this model to select the best methodology for a given project.

4.2 The Balanced Scorecard Model

The Balanced Scorecard (BSC) (Figure 1) is a recent approach to strategic management and performance measurement. It was issued within the industrial community and has gained considerable interest. The Balanced Scorecard groups similar types of measures into sets regarded as perspectives (Figure 1). The BSC model of Kaplan and Norton [24] defines four complementary perspectives which can evaluate the capability of a company: financial, customer, internal business process, and learning and growth. These four perspectives balance short and long-term objectives. They focus on both financial outcomes and long-term competitive capabilities. The four perspectives are detailed here [25]:

- **Customer Perspective** - Many companies aim to be number one in delivering value to customers. Customers may show many concerns: time, quality, performance and cost.
- **Internal Business Process -** Managers need to focus on those critical internal operations that enable them to satisfy customer needs.
- **Financial Perspective -** Typical financial goals have to do with profitability, measured for example, by operating income, return-on-capital-employed and economic value-added.
- **Learning And Growth -** Intense global competition requires that companies make continual improvements to their existing products and processes and have the ability to introduce entirely new products with expanded capabilities.



How Do We Look To Shareholders ?
Financial Perspective
Goal
Measures
What Must We Excel At ?
Customer Perspective
Goal
Measures
Internal Business Perspective
Goal
Measures
How Do Customers See Us ?
Innovation and Learning Perspective
Goal
Measures
Can We Continue To Improve and Create Value

Figure 1 : *The Balanced Scorecard*

4.3 The Balanced Scorecard as a Driver for Selection

Beyond its normal use as a tool for strategic management and a balanced business growth, the Balanced Scorecard can be used to compare software development methodologies. With its multi-criteria performance indicators, we can evaluate software methodologies against the four perspectives defined by the BSC. For a single project, a software company may be willing to prioritize one or more perspectives more than the other ones. By classifying the different factors affecting software development into the four aspects of the BSC, we get useful guidance on the elements to be considered to achieve the required perspectives and satisfy the company needs. Figure 2 presents such a classification. It refers to the value that we might get from using a methodology. Characteristics for the project, organization, and methodology were distributed over the four perspectives showing at the same time common points between them. The different characteristics are grouped as follows:

- **Internal business process:** includes adoption and tailorability, team size, tools support, innovativeness and domain of application.
- **Financial:** includes costs for tools support and costs for training for team members, return on investment and available resources.
- **Learning and growth:** is reflected in factors such as software process improvement capabilities, quality achieved, dynamism, and return on investment.
- **Customer perspective:** factors to improve the image of the company. Software quality and process maturity are the main factors affecting it.

4.4 General Recommendations

Selecting a methodology requires gathering enough information on the project, the organization maintaining the project, and available methodologies and their scope of use. We can then map our project with the suitable methodology. Organization's related information should be investigated first: software process capability, project management support, skills, knowledge and willingness of people, discipline or culture, orientation of the team manager, etc. The next step should address the project elements: quality required, innovativeness, domain of application, criticality, required resources, etc. At this point, with the help of the provided framework, the team manager can make his choice dependent on the objectives and preferences of the project. He or she will adopt a methodology that contributes most to the achievement of the perspectives which conform with the fixed objectives and then adapt it if necessary to fit the organization's needs.

5 Case Study

5.1 Introduction

With the guidelines and considerations outlined in section 3 and 4, we conducted a case study at a software development company to demonstrate the applicability of the provided framework. We will describe first the case organization: the development team and its stakeholders and the existing processes within the organization. We will then describe the product to be developed: its functionality, quality required and its criticality to both the company and the customer. With reference to the different factors analyzed in section 3, we recommend one of the four methodologies studied to be adopted for this project. Arguments will be provided to justify our choice.

5.2 The Organization

Macro Data World or « MDWorld » is a software development company specialized in Decision Support Systems (DSS). MDWorld was created in 2000 by a group of software engineers in association with MCS Company (Mediterranean Consult and Service Company), the leader of consulting in Algeria. MDWorld was created to complement and extend MCS Company activities. Its aim was to furnish efficient solutions for its customers, such as software development and the study and evaluation of existing information systems. MDW has been developing software since 2000 when the development team consisted of 4 developers. Since then the development team has grown up and now has 7 people, 5 of whom are directly involved in software development. The development team is made up of a group of experienced permanent staff that have been working together since 2000. This developed a good sense of communication within the group and allowed them to share knowledge in an efficient way. The five-year period of teamwork allowed the company to establish some development and managerial processes but there was no formal description of activities to be done with no history of previous projects. Development activities are held in a four-room flat which is also used as a headquarter for the administration of the company.

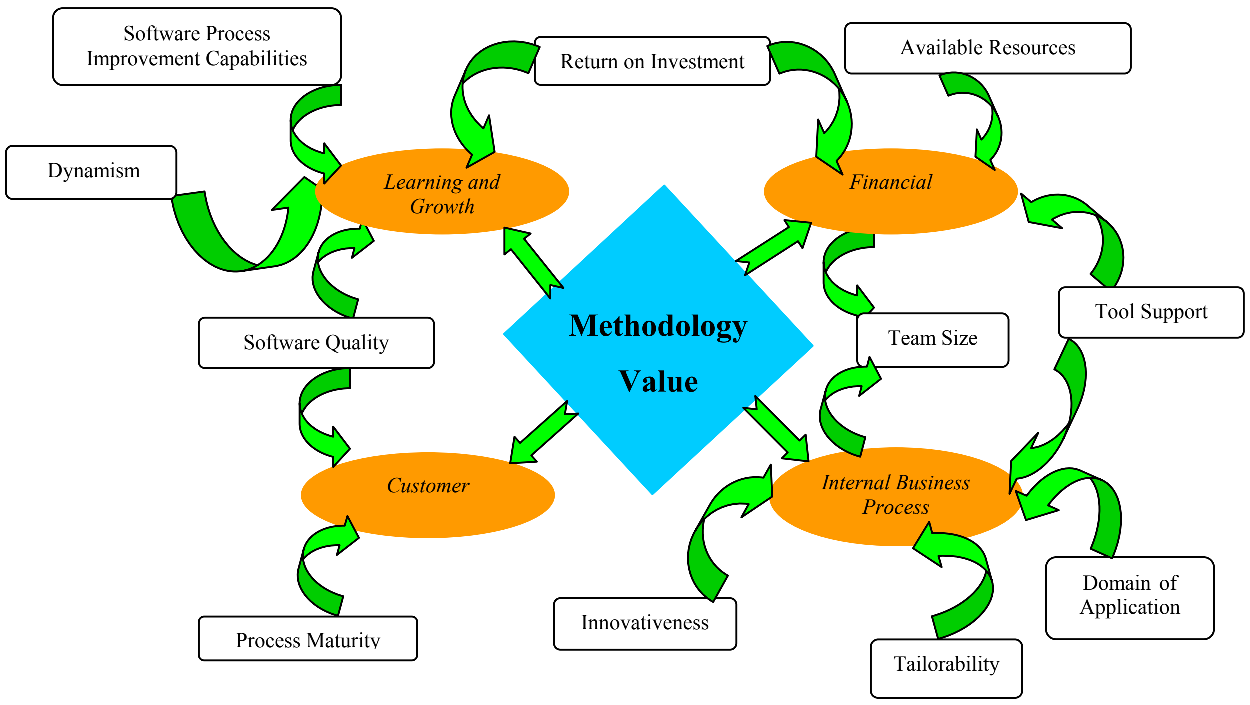


Figure 2: *Methodology Value*

5.3 The Project

The mission of the project is to create a business intelligence analysis tool based on Data-Warehousing techniques to help companies to store company-related information and to provide in-depth analyses of data gathered whenever necessary, in order to support management decision making processes. The product is to be called Maestro. It is to be developed as a COTS product. It is customizable and needs to be configured before being used by companies. The Dot Net framework was chosen to develop the product. The development team used to Delphi environment and decided now to move to Microsoft's recent technology. Team members have little experience with Dot Net technology. Special training is required to adapt with the new environment. The company planned a one-month training period for all team members and allocated a special budget for this. The product is supposed to perform the following functions:

- Consolidate and centralize all available data.
- Provide a dashboard view of the business.

- Use different models for data analysis, and
- Ensure information broadcasting all over the company.

The company has already assessed its software development process and found it as being equivalent to level one of the CMM. The group established a program to increase the maturity level at least to level two. Last, and for special obligations, the project must be finished within three months. Respecting the fixed deadline is also highly required.

5.4 Discussion

5.4.1 Project Factors

Regarding the application domain, we can eliminate Cleanroom which is not a good choice to develop a business application. The project is critical to the company due to the rigid deadline imposed by the managerial staff. Criticality favors plan-driven methodologies (RUP) rather than agile ones (XP and OS). As for quality, the product does not require a high level of quality. Quality has little impact on our choice since the three remaining methodologies can ensure the required level of quality. Concerning innovativeness, MDW has already dealt with many similar projects. The present product is considered as an improvement to an old system. XP deals better with legacy code and promotes better results.

5.4.2 Methodology Factors

The company cannot afford tool support required by RUP. Some Open Source tools are used instead. XP and Open Source do not require such resources. Return on investment is critical to the well-being of the company. There is an urgent need to recover money being invested since 2000. XP offers a short-term benefit while RUP and Open Source promise a long-term benefit. The company cannot wait longer to see the benefits of its investment. MDW management insists on achieving a higher level of the CMM. RUP and XP offer more possibilities than Open Source Development. Adoption prioritizes XP and with a minor degree RUP. Good communication between team members and the open environment inside the company give XP more acceptance and make it easier to adopt. RUP is rigid and resource-hungry while Open Source requires a lot of time and has little acceptance within the company considering it as a chaotic process.

5.4.3 Organization Factors

A seven members-team is an ideal size of an XP project and fits well the development of the present product. Open Source is also possible but we do not have any indicator as to the team yield. A smaller version of RUP is also possible after a customization process. Dynamism highly favors XP and Open Source in the case of MDW. The small budget allocated to the project favors XP and Open Source and makes the project accomplishment uncertain using RUP.

5.5 Conclusion

Achieving a higher level of maturity is a strategic goal for MDW. It will serve in the future as a marketing device to remain competitive and win contracts. On the other hand, there is an urgent need to recover its investments as soon as possible. These are the two main objectives that the company wants to achieve. The first objective ties with customer perspective represented by process maturity and software quality factors while the second one ties with financial perspective seen as maximizing return on investment by minimizing expenses. Open Source does not satisfy the first expectation and is not therefore recommended. Regarding financial aspects, the need for a short payback period for the investment and the small allocated budget for the project make XP the best choice. As for process improvement capabilities, both XP and RUP offer higher maturity levels which comply with the fixed objectives. The rest of the factors (adoption and tailorability, innovativeness, domain, quality, criticality, and team size) have little influence on the choice because they are dealt with nearly in the same way by both methodologies XP and RUP. The precedent factors make XP the best choice for the development of the studied project.

6. Conclusion and Suggestions

This Study deals with comparing and selecting software development methodologies. The need for comparing software development methodologies arises from the perplexity that faces organizations and project managers when selecting a methodology for their projects. We introduced a new approach for selection based on the Balanced Scorecard, an industrial model which proved its consistency over the time. In addition to guidance provided by the proposed model in selecting methodologies, its interest lies in that it takes into consideration the future developments and projects of the company and does not limit the company's vision to a single project. Methodologies for future projects will be chosen in the same way, ensuring continuity in achieving the desired objectives. The environment created by each project will complement what has already been done. However, this model may be improved by using additional perspectives in the BSC model to suit software development. In fact, software development has special characteristics that make it differ from industrial projects. An in-depth study is needed to discover the other possible perspectives to be taken into account. The framework used, composed of the different characteristics of the project, organization, and methodology is not exhaustive but shows only the most important factors of these three elements. Further study will allow us to find more about this point. Using the BSC model as a starting point will allow to identify other relevant characteristics and therefore to enrich the framework with more factors that may affect the choice. The same thing is to be said on the studied methodologies. Introducing more methodologies will bring more diversity and give more credibility to the study. The Balanced Scorecard is one of many industrial models that have been successfully adopted since decades. Software development has common features with the industrial community and has already borrowed many techniques and tools from it. In order to achieve better performance and to bypass the current crisis, the software community may investigate for industrial models or standards to be used for improving the way software is developed.

References

[1] – Robert C. Glass, “Process Diversity and Computing Old Wives’/Husbands’ Tale,” IEEE Software, July 2000

[2] – Mikael Lindvall and Ioana Rus, “Process Diversity in Software Development,” IEEE Software, July/August 2000

[3] - B. Henderson-Sellers, G. Collins, R. Dué and I. Graham, “A Qualitative Comparison of two Processes for Object-Oriented Software Development,” Information and Software Technology, 43 (2001) 12

[4] - B. Henderson-Sellers and C. Gonzalez-Perez, “A Comparison of four Process Metamodels and the Creation of a New Generic Standard,” Information and Software Technology, 47 - 2005

[5] – Frina Albertyn, “Comparing Possible E-Commerce Processes,” Proceeding of the 16th Annual NACCQ, Palmerston North, New Zelanda, July - 2003

[6] - Philippe Kruchten, *The Rational Unified Process: An Introduction*. Addison Wesley, 2003

[7] - Per Kroll, Philippe Kruchten, *Rational Unified Process Made Easy: A Practitioner’s Guide to the RUP*. Addison Wesley, 2003

[8] - Kent Benck *Extreme Programming Explained - Embrace Change*. Addison Wesley, 1999

[9] - Richard C. Linger and Carmen J. Trammell ,“Cleanroom Software Engineering Reference Model - Version 1.0,“ Technical Report CMU/SEI-96-TR-022 ESC-TR-96-022, November 1996

[10] - R. Linger, “Cleanroom Process Model,” IEEE Software, March 1994

[11] – Jason Charvat, *Project Management Methodologies: Selecting, Implementing, and Supporting Methodologies and Processes for Projects*. John Wiley and Sons, 2003

[12] – R. C. Linger "Cleanroom Software Engineering for Zero-Defect Software," Proceedings of the 15th International Conference on Software Engineering, Computer Society Press, May 1999

[13] – Scott W. Ambler, John Nalbone, Michael J. VizdosThe , *Enterprise Unified Process : Extending the Rational Unified Process.* Prentice Hall PTR, 2005

[14] – M. D. Deck, "Cleanroom Software Engineering and 'Old Code' -- Overcoming Process Improvement Barriers," Proceeding of the Pacific Northwest Software Quality Conference, October, 1995

[15] - Alfonso Fuggetta, “Open Source Software - An Evaluation,” Journal of Systems and Software, Volume 66, Issue 1, April 2003

[16] - Barry Boehm and Richard Turner, *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison Wesley, 2003

[17] - Juhani Warstaa and Pekka Abrahamsson “Is Open Source Software Development Essentially an Agile Method?” 3rd Workshop on Open Source, International Conference on Software Engineering, Portland, Oregon, May 3-11, 2003

[18] – Lisandra V. Manzony and Roberto T. Price “Identifying Extensions Required by RUP to Comply with CMM Levels 2 and 3,” IEEE Transactions on Software Engineering, February 2003

[19] – Mark C. Paulk, “Extreme Programming from a CMM Perspective,” IEEE Software, November/December 2001

[20] - Richard C. Linger, Mark C. Paulk and Carmen J. Trammel, “Cleanroom Software Engineering Implementation of the Capability Maturity Model for Software,” Software Engineering Institute, December, 1996

[21] - "Open Source Maturity Model," Available at: http://www.navicasoft.com/pages/osmm.htm

[22] – W. Bleek, M. Finck and B. Pape, “Towards an Open Source Development Process –Evaluating the Migration to an Open Source Project by Means of the Capability Maturity Model,” Proceedings of the First International Conference on Open Source Systems (OSS 2005), 11-15 July 2005, Genua, Italy

[23] P. Hausler, R. Linger and C. Trammell, “Adopting Cleanroom Software Engineering with a Phased Approach,” IBM Systems Journal, 33[1] 1994

[24] – Robert S. Kaplan and David P. Norton, *The Balanced Scorecard: Translating Strategy into Action*. Harvard Business School Press, Boston, Massachusets – 1996

[25] – Robert S. Kaplan and David P. Norton, “The Balanced Scorecard – Measures that Drive Performance," Harvard Business Review, January-February 1992