A Non-Intrusive Process to Software Engineering Decision Support focused on increasing the Quality of Software Development

Everton Gomede and Rodolfo M. Barros
Computer Science Department
State University of Londrina, UEL
Londrina, Brazil
evertongomede@gmail.com, rodolfo@uel.br

***Abstract* — The lack of quality in the production process of software development isn't attributed only to the techniques and technologies, but also to the lack of process that *management decisions*. Thus, this paper presents a process model for *Software Engineering Decision Support* focused on improving the quality of software development. Its preparation was based on areas and expected results of the process *Decision Management* present in the *Reference Model for Brazilian Software Process Improvement* (MR-MPS)[1]. In order to contribute to its understanding and use, it is presented a comparative study with other models present in the literature and identifies its benefits and problems with an application in two software development projects. The result of this process was a 78% reduction in rework and a 22% increase in performance of the team.**

Keywords - Decision Support; Analytic Hierarchy Process; Historical Database; Increase Quality of Software Development.

I. INTRODUCTION

During the software development lifecycle we can find a set of decisions that should be taken in order to *increase* product quality and / or respect any project restrictions imposed [1, 3, 6, 14]. Some of these restrictions can be seen in Fig. 1. But (*i*) what are the decisions that must be taken throughout the software development lifecycle? (*ii*) How these decisions affect the later stages and final product quality? (*iii*) How to make structured and tracked decisions throughout the software development lifecycle? (*iv*) And how to make these decisions not intrusive to the existing software development process?

Figure 1. Some restrictions that must be balanced in a project [1].

We will examine these issues in greater depth starting from the issue (*i*):

[1] Modelo de Referência para Melhoria do Software Brasileiro (MR-MPS)

A. *What are the decisions that must be taken throughout the software development lifecycle?*

Consider a software development process such as *Rational Unified Process* (RUP) [2] shown in Fig. 2. Several decisions must be made along each disciplines and iterations. For instance, on the discipline "Business Modeling" decisions as (*i*) which processes are the most urgent? (*ii*) What processes are at greatest risk? (*iii*) What are the core and support processes? And others may emerge early in the software development process. Its results will affect the other phases of the process [1]. This leads us to the second issue:

B. *How these decisions affect the later stages and final product quality?*

The next steps of the process will be affected since they use up the results of previous decisions to plan their executions [1]. Regarding the quality of the final product, the result will be a *very strong* relationship to the quality of the process [3]. Since decisions were made erroneous so there is a greater probability of final product to be a poor quality.

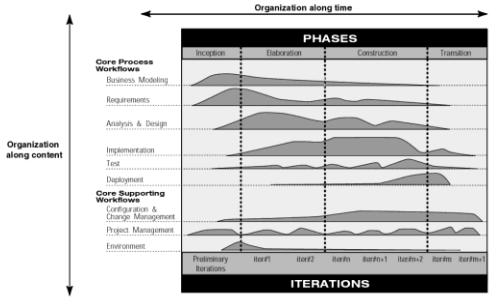

Figure 2. The Rational Unified Process (RUP)

C. *How to make structured and tracked decisions throughout the software development lifecycle?*

In an engineering work where most decisions are techniques [5], it should be structured and stored in a *Historical Database* (HDB). The decisions created and stored in HDB can be accessed and / or reused in the future, making the HDB in an organizational asset [1]. Last but not least:

D. How to make these decisions not intrusive to the existing software development process?

Despite the engineering, software projects are creative [6]. Extra *bureaucracy* can reduce the creativity of developers and / or create unnecessary overhead.

Considering what was previously exposed, the aim of this work is to present a model of process focused on the increase of the quality of the software development process. For its elaboration we based on the expected areas and results of the *Decision Management Process* presents in the *C* maturity level of MR-MPS [14]. For this we propose a non-intrusive process to support decision making in software engineering (NIPSEDS) using the method *Analytic Hierarchy Process*[2] (AHP) and a Historical Database (HDB) to address the issues *A*, *B*, *C* and *D*.

This article is divided in six elementary sections, including this introduction. In Section 2 we presented the related work and theory. In Section 3 we presented the process model to *Software Engineering Decision Support*. Section 4 we presented the validation of model through a case study. Section 5 we presented the results of the research. Finally, Section 6 we presented the conclusions and suggestions for future works.

II. THEORY

A. Analytic Hierarchy Process

The *Analytic Hierarchy Process* (AHP) was first proposed by Thomas L. Saaty [15] and its main characteristic is the *pairwise comparison* which consists of a *hierarchy of criteria and alternatives*. It is often used to analyze problems of decision-making multi-criteria. By using AHP, the structure of the problem must be decomposed into a hierarchy.

A hierarchy is a specific system based on the assumption that the entities can be grouped into disjoint sets with a group of entities which affects the other ones [15]. Pairwise comparison is an important component of the AHP. Two criteria are compared using a nine-point scale, where one (*1*) means "equal" importance, three (*3*) is "low" importance, five (*5*) "indicates" clearly "superior", seven (*7*) is "very" important and nine (*9*) denotes "extremely" important. With pair numbers being used to indicate intermediate values, if necessary. If there are *n* criteria to consider, *n(n-1)/2* comparisons of pairs had to be done. Thereafter, the reciprocal *n*x*n* matrix is constructed and weights are then obtained [11, 12].

The consistency of pair comparison matrix needs to be verified by means of the indexes: *Consistency Index* (CI) and *Consistency Rate* (CR). They are defined in equation (1) and (2) with λ_{max} being the principal value (Eigen) and *Random Index* (RI) is as shown in Table I. For consistency, CI and CR *must* be less than 0.1 for the AHP analysis is considerate acceptable [11, 12].

$$C.I. = (\lambda_{max} - n) / (n - 1) \quad (1)$$

$$C.R. = C.I. / R.I. \quad (2)$$

[2] Implemented through *Expert Choice*, http://expertchoice.com/

TABLE I. RANDON INDEX

n	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
R.I.	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49

B. Related Work

In addition to recommendations of the MR-MPS guide [14], we found in literature some works that address issues related to management decisions during the software development lifecycle. In [7] the authors integrating the *Technique for Order of Preference by Similarity to Ideal Solution* (TOPSIS) and AHP into *Goal-Oriented Requirements Engineering* (GORE) in a *Decision Support System* (DSS) to produce a metric of choice among the best alternatives. However, this paper addresses only the *initial phase* of the project. In [8] the authors use a group decision technique only to the requirements phase. In [9] the authors use data mining techniques to software engineering decisions. This adds an *overhead* to the process. In [10] the authors document the decisions made throughout the software development lifecycle but without the concern in *structuring* decisions. This work presents a historical database. Table II compares related work with this model process.

TABLE II. RELATED WORK COMPARED WITH THIS MODEL PROCESS

Criteria	**Related Work**				
	[7]	[8]	[9]	[10]	This
Structured Decision	●	●	●		●
Decision Traced			●		●
During the Lifecycle				●	●
Historical Database			●	●	●
Non-Intrusive Process	●	○			●

● Strongly cares to ○ partially care to and without symbol don´t attends. These criteria support areas and expected results (from GDE1 to GDE7) of Decision Management Process of MR-MPS [14].

C. Reference Model for the Brazilian Software Improvement Process[3]

Developed in 2003 by the SOFTEX[4] as part of the MPS.Br[5] program, the MR-MPS consists of a reference model with the definition of prerequisites for the improvement of the quality of the software process. Besides it, the program is composed by an Assessment Method (MA-MPS) and a Business Model (MN-MPS), each one of them described by guides and/or document models.

In accordance with Capability Maturity Model Integration for Development (CMMI-DEV) and following the described headlines in its main program, this model was divided into seven maturity levels. These levels define steps to improvement processes in the organization [14]. Moreover, this division aims to enable its implementation and assessment in micro, small and medium enterprises.

These maturity levels are composed by processes which define what the expected results are, and capabilities which express its institutionalization level and implementation in the organization. Thus, it is noteworthy that the development

[3] Modelo de Referência para Melhoria do Software Brasileiro (MR-MPS)

[4] Associação para Promoção da Excelência do Software Brasileiro

[5] Programa para Melhoria do Processo do Software Brasileiro (MPS.Br).

among these levels happens cumulatively and only when all demands were found.

III. NON-INTRUSIVE PROCESS TO SOFTWARE ENGINEERING DECISION SUPPORT (NIPSEDS)

To characterize the proposed process we divided it into (*i*) Activities, (*ii*) Roles, (*iii*) Tools & Techniques and (*iv*) Inputs and Outputs.

A. *Activities*

We grouped the activities of process in groups identified as (*i*) Structure Decision, (*ii*) Make Decision (*iii*) Store Decision and (*iv*) Publish Decision. This division aims to facilitate understanding and enable semantics view related to the actors. These groups are based on *C* level of MR-MPS [14]. Fig. 3 shows these activities.

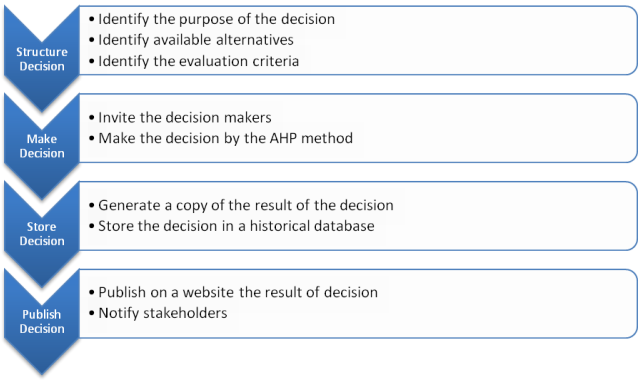

Figure 3. Process Groups of the Non-Intrusive Process to Software Engineering Decision Support (NIPSEDS)

This support process can be executed at any discipline or RUP iteration (Fig. 2). RUP is a process used by public university (Section 4) of this case study. The NIPSEDS can be applied to any process of software development. Fig. 4 holds a more detailed model.

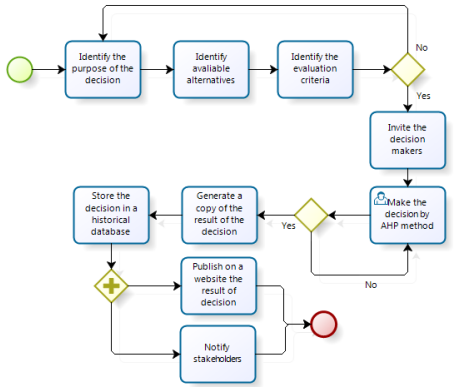

Figure 4. The Non-Intrusive Process to Software Engineering Decision Support (NIPSEDS)

B. *Roles*

The *roles* used in the process are two: (*i*) *Project Manager* (or *Scrum Master*) and (*ii*) *Decision-Makers* (which can be developers, database administrators, architects, testers, business analysts, and others). Fig. 5 shows these roles and their relationship with the activities.

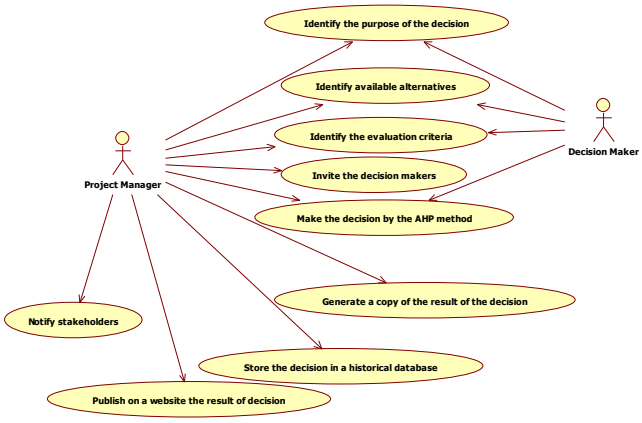

Figure 5. Roles of The Non-Intrusive Process to Software Engineering Decision Support (NIPSEDS)

Note that the *Project Manager* participates in all process activities. This is important to have an "Owner" of the process being responsible for ensuring the use of it and its constant improvement. The role of *Project Manager* was chosen to represent someone with administrative and managerial responsibilities for the project and not only with *technical responsibilities*.

C. *Tools & Techniques*

The AHP was the technique used to structure the decision. Further details and examples of how to use it can be seen in [11, 12]. For the tool we used the *Expert Choice*. It´s important to note that in this process is possible to use tools and techniques adapted to the software development process of the organization. The AHP technique comprises three activities of the group process *Structure Decision*:

- *Identify the purpose of the decision*. This activity seeks to identify the final goal of the decision, i.e., what we intend to achieve. As obvious as it may seem, this is not always trivial.
- *Identify available alternatives*. Identifying alternatives consists basically of an investigation process. The alternatives available are not always by the team known and / or have been used in the past by the organization. The important thing here is to research and rank the possible options that can be used for decision making.
- *Identify the evaluation criteria*. The criteria are the attributes that the alternatives listed must be compared. These criteria may be conflicting or mutually exclusionary. The AHP helps prioritize these criteria into a hierarchy [11, 12].

D. *Inputs and Outputs*

The process inputs are (*i*) the decision objective, (*ii*) a set of alternatives, (*iii*) a set of criteria, (*iv*) the stakeholders and a (*v*)

method to assist in structuring the decision (in the case AHP). The outputs are (*i*) the decision result and (*ii*) the decision documentation, thus creating an organizational memory [1].

These inputs and outputs are important to the creation of *Historical Database* (HDB). This artifact can be considered as an organizational asset [1], since it stored the decisions made throughout the lifecycle and to allow that future decisions are based on a set of criteria that are always feedback. Fig. 6 shows a HDB class diagram.

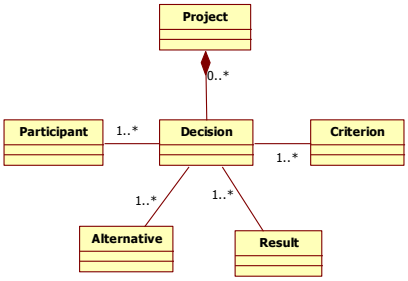

Figure 6. The Historical Database (HDB) class diagram

IV. VALIDATION

The research methodology used in this article was a case study. According to Yin [13], case studies offer an empirical research that investigates a contemporary phenomenon and offers researchers an object of applied study in its natural context. And, in addition, new facts and research issues about this environment can be identified [13].

In order to work on the case study, we selected a project of a software factory in a public university. Their teams were composed by undergraduate and master's students. Because of this, the organization suffers with the seasonality issues in periods of academic activity, lack of commitment, interest and a low rate of productivity in its members. Another problem of this organization is the lack of a process of preservation of intellectual capital generated during the projects.

During two projects with 6 iterations of 15 days each, we apply the NIPSEDS and 3 variables were collected (*i*) *Rework Index*, (*ii*) *Structured Decision* and (*iii*) *Performance Index*. To illustrate the NIPSEDS, one structured decision will be presented below separated by the process groups. This decision was performed in the second iteration of the first project and is intended to decide which persistence framework to use.

A. *Structure Decision*

The outputs of these process group activities are summarized in Fig. 7. The *alternatives* are (*i*) Entity Enterprise Java Beans[6], (*ii*) Hibernate[7], (*iii*) Java Persistence API[8] and (*iv*) TopLink[9]. These are some persistence framework to java software development. It is important to note that all *decision elements* (goal, criteria and alternatives) so collected by the team.

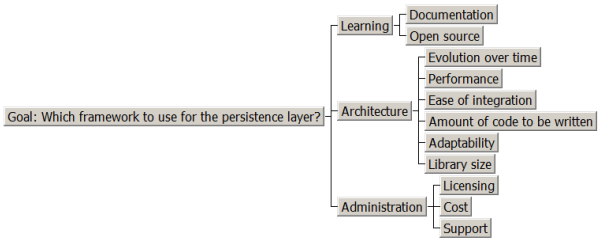

Figure 7. Criteria hierarchy ("*Struture Decision*" activities output)

B. *Make Decision*

With the established hierarchy made up some iteration where each participant reported their preference about the criteria and alternatives [11, 12]. The result of these preferences can be viewed in Fig. 8.

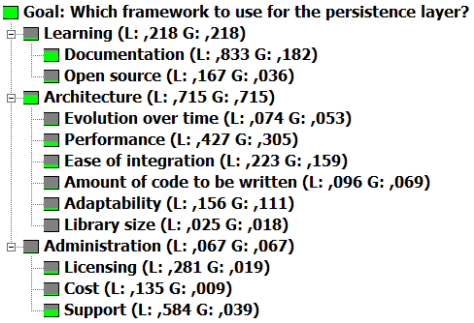

Figure 8. Hierarchy with the preferences result (more details in [11, 12])

After consensus about the choice, the outcome of the decision can now be display. Fig. 9 shows decision results.

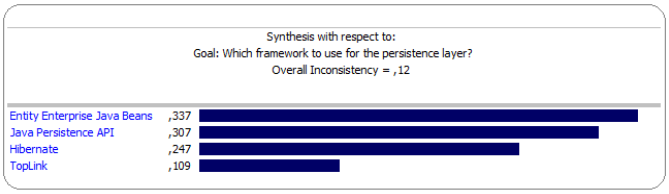

Figure 9. Decision results (represents a consensus about the choice)

The *Expert Choice* allows *different* analyzes about the decision taken. Two of them can be seen in Figures 10 and 11 respectively. Fig. 10 shows the sensitivity of alternative groups of criteria. Fig. 11 shows the result of adherence with relation to the criteria.

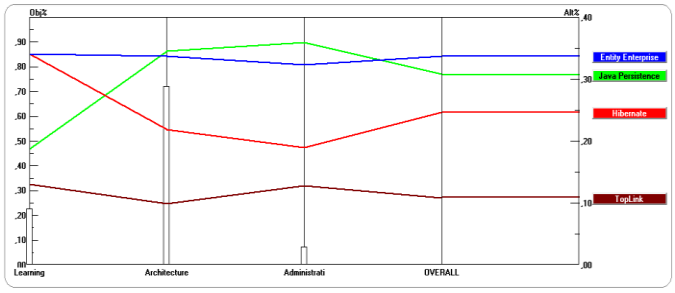

Figure 10. Alternatives sensibility of on criteria group

[6] http://docs.oracle.com/cd/E16764_01/web.1111/e13719/toc.htm
[7] http://www.hibernate.org/
[8] http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html
[9] http://www.oracle.com/technetwork/middleware/toplink/overview/index.html

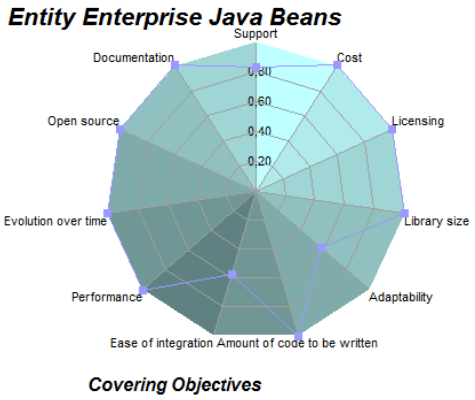

Figure 11. Adherence with relation to the criteria (note *coverage* of ≃72% regarding the the objective criteria)

C. Store Decision

After consensus about the choice, the outcome of the decision can now be stored. Table III shows an example of stored result.

TABLE III. EXEMPLE OF STORED RESULT

Attribute	***Value***
Date & Time	18/10/2012 – 10:50
Goal	*Which framework to use for the persistence layer?*
Alternatives	1. Entity Enterprise Java Beans 2. Hibernate 3. Java Persistence API 4. TopLink
Criteria	*1. Learning* 1.1. Documentation 1.2. Open source *2. Architecture* 2.1. Evolution over time 2.2. Performance 2.3. Ease of integration 2.4. Amount of code to be written 2.5. Adaptability 2.6. Library size *3. Administration* 3.1. Licensing 3.2. Cost 3.3. Support
Result	1. Entity Enterprise Java Beans
Decision Makers	Bill Mark Steve

These data were stored in the structure shown in Fig. 6 (Section 3 *D*). Artifacts such as images and Portable Document Format (PDF) can be annexed increase the quality of Historical Database (HDB). The Decision Maker's names were changed for confidentialy questions.

D. Publish Decision

The published result can be seen in Fig. 12. Effective communication creates a bridge between diverse stakeholders who may have different culture and organizational backgrounds, different levels of expertise, and different perspectives and interests, which impact or have an influence upon the project execution or outcome [1].

Project Lambda / Analysis and Design

Decision about the persistence framework

Attribute	***Value***
Date & Time	10/03/2013 – 10:50
Goal	Which framework to use for the persistence layer?
Alternatives	1. Entity Enterprise Java Beans 2. Hibernate 3. Java Persistence API 4. TopLink
Criteria	*1. Learning* 1.1. Documentation 1.2. Open source *2. Architecture* 2.1. Evolution over time 2.2. Performance 2.3. Ease of integration 2.4. Amount of code to be written 2.5. Adaptability 2.6. Library size *3. Administration* 3.1. Licensing 3.2. Cost 3.3. Support
Result	1. Entity Enterprise Java Beans
Decision Makers	Bill Mark Steve

Figure 12. Published decision in project´ website

V. RESULTS & ANALYSIS

In order to validate the process model, some performance indicators for information and data collection were defined and applied (Section 4). Through the analysis of these sources, it was possible to identify *advantages* and *limitations* of NIPSEDS. Afterwards, the results obtained with this research are described. The first indicator shows the variation in the rate of rework. This is because high levels of rework were presented as major problems during the development of a project.

Through decisions made throughout the project, we have tried to reduce the number of rework. Thus, solving the rework, this metric helps the project manager to identify the level of effectiveness of decisions. Fig. 13 shows this indicator.

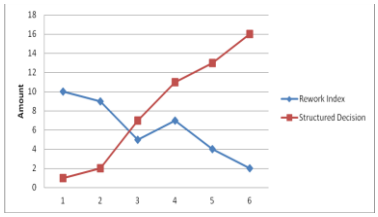

Figure 13. Rework Index vs. Structured Decision

As can be observed in Fig. 13, the structured decisions have strong relationship with the decreasing of rework. After the implementation of the framework, this fact is evidenced by the *decrease* in ≃78% (average) of this index in Projects, that research contributes to the quality in the development process. Besides this, there was an improvement on the perfomance index of members by structuring decisions during project, it is also important for improving quality in the development process. This happens because the effectiveness of the performance actually contributes to the effectiveness of the members. Thus, by sctructuring decisions, seeks to empower and qualify them so they can increase this indicator. Furthermore, through this measure, makes it possible to project manager to analyze the performance of its members and, if necessary, take steps to improve them. Fig. 14 has the graphics prepared for analysis of this index.

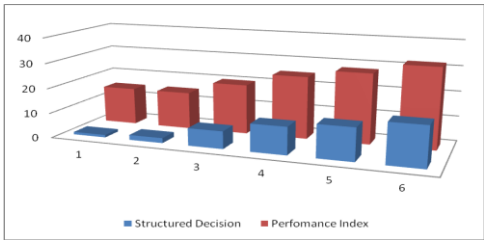

Figure 14. Performance Index vs. Structured Decision

Besides the rate of rework the structuring decision also maintains a strong relationship with the improvement in the performance index of members of the team. This fact strongly evidenced by analyzing the graphs shown in Fig. 14. Through them, it is noted that with the number of structured decision, implemented by the framework, an *increase* of ≃22% in performance of the members. And that contributes not only to meet the deadline and measurement of the team, but also to improve the quality of coding, and especially the ease of maintenance. Thus, in a general way, through the analysis of the performance indicators and collected information, the following advantages were identified:

- *Increased understanding of decisions*: With the structuring of decisions, the understanding of the problem to be solved increases. This is reflected in the later stages where decisions of the past can be retrieved and validated;
- *Improvement in the development process*: The ongoing process of analysis and monitoring ensured that the best options were selected for each objective;
- *Improvement in choice of criteria and/or alternatives*: These activities are improved by the selection and utilization of criteria, alternatives and objectives stored in the *Historical Database* (HDB);
- *Increase of the organizational memory*: The storage of experiences, estimates, knowledge and performance of team's members during the development of the projects, in the suggested HDB, has as objective to keep this information available in the beginning of every project in order to facilitate the future decisions.

Moreover, through continuous monitoring of performance and decision aspects in software projects, it can be stated that the mentioned advantages have contributed significantly for the decrease of its rework and for the increase in performance and improvement of its activities development. All these factors, besides contributing significantly for the application of the process model, also collaborate to the establishment of an asset within the organization.

VI. Conclusions and Future Works

Analyzing the results obtained during the case study development, we can evaluate the success in the implementation of the process model. It is highlighted, mostly, the increase in motivation of members and their performance, resulting in a significantly improvement in its development process and decrease rework.

Thus, focused on *increase* the *quality* of software development process, the process model presented was developed to attend, provide and add more value to process used by organization through planning and continuous structuring decisions. One possible limitation of this work is the need for a certain level of maturity in software development. *C* level according to MR-MPS [14]. In this case study the responsibility to lead the process was delegated to the most *experienced* organization member (*Project Manager* or *Scrum Master*). Finally, as presented in Table II, this process model differs from other existing process models in literature and can be applied in a *non-intrusive* way.

As future work we intend to analyze the relationship between times spent on decisions versus the time saved with rework. This has an *economic* objective related to software development.

References

[1] PMI, A Guide to the Project Management Body of Knowledge (PMBOK guide), Fifth Edition, Project Management Institute, Inc, 2013.

[2] P. Kruchten, The rational unified process: an introduction, ser. The Addison-Wesley object technology series. Addison-Wesley, 2004.

[3] Lavallée, M., & Robillard, P, "The impacts of software process improvement on developers: a systematic review". International Conference on Software, 113–122, 2012.

[4] Xuan S., "A Novel Kind of Decision of Weight of Multi-attribute Decision-Making Model Based on Bayesian Networks, Business and Information Management", 2008. ISBIM '08. International Seminar on , vol.2, no., pp.30,33, 19-19 Dec. 2008.

[5] Colwell, B., "Engineering decisions", Computer, vol.36, no.8, pp.9,11, Aug. 2003.

[6] Yang F., Zhang W., "Exploration and practice of constructing creative engineering laboratory on software development", Computer Science & Education (ICCSE), 2012 7th International Conference on , vol., no., pp.1597,1601, 14-17 July 2012.

[7] Vinay, S., Aithal, S. and Sudhakara, G., "Integrating TOPSIS and AHP into GORE Decision", International Journal of Computer Applications (0975 – 8887) Volume 56– No.17, October 2012.

[8] Felfernig A., Zehentner C., Ninaus G., Grabner H., Maalej W., Pagano D., Weninger L., and Reinfrank F., "Group decision support for requirements negotiation". In Proceedings of the 19th international conference on Advances in User Modeling (UMAP'11), Springer-Verlag, Berlin, Heidelberg, 105-116. 2011.

[9] Hassan, A., and Xie, T., "Software intelligence: the future of mining software engineering data". SDP workshop on Future of software engineering, 161–165. 2010.

[10] Lewis, T., Spillman, R., and Alsawwaf, M., "A software engineering approach to the documentation and development of an international decision support system". Journal of Computing Sciences. 2010.

[11] Gomede, E., Barros, R. M., "Utilizando o Método Analytic Hierarchy Process (AHP) para Priorização de Serviços de TI: Um Estudo de Caso." In: VIII Simpósio Brasileiro de Sistemas de Informação, São Paulo. VIII Simpósio Brasileiro de Sistemas de Informação, p. 408-419. v. 1. 2012.

[12] Gomede, E., Proenca JR., M. L. and Barros, R. M., "Networks Baselines And Analytic Hierarchy Process: An Approach To Strategic Decisions." In: IADIS International Conference Applied Computing, 2012, Madri. IADIS International Conference Applied Computing, p. 34-41. 2012.

[13] Yin, R. K, Case Study Research: Design and Method, Third Edition, Applied Social Research Methods Series, Sage Publications, Inc, 2002.

[14] MR-MPS (Modelo de Referência para Melhoria de Processo do Software Brasileiro). Associação para Promoção da Excelência do Software Brasileiro, December, 2012.

[15] Saaty, T. L, The Analytic Hierarchy Process. New York: McGraw-Hill International. 1980.