Journal of Technology Management & Innovation

Received July 1, 2008/ Accepted August 1, 2008

AN EVALUATION APPROACH BASED ON THE PROBLEM-BASED LEARNING IN A SOFTWARE ENGINEERING MASTER COURSE

Ana Paula Cavalcanti[1,2], Simone Santos[1,2], Maria da Conceição Moraes[1,2], Jones Albuquerque[1,2], Silvio Meira[1,2]

[1]C.E.S.A.R.edu – Educational Unit of Recife Center for Advanced Studies and Systems
Rua Bione, nº 220, Cais do Apolo 50.0303-90, Recife - PE, Brazil –Phone number: +55 (81) 3425 – 4700

[2]CIn - Informatics Center from Federal University of Pernambuco (UFPE)
Caixa Postal 7851, Cidade Universitária – 50.732-970 – Recife – PE – Brazil –Phone number: +55 (81) 2126 - 8430
{ana.paula, simone.santos, ceca.moraes, jones.albuquerque, silvio}@cesar.org.br

Abstract

The effectiveness of a Master's Program Course can only be appraised with the support of an efficient and accurate evaluation program. Taking the Problem Based Learning, as a reference to implement practical and real problems on a class of master students, and the traditional methods that focuses on formal exams as methodology, a method for evaluation was defined to contemplate both scopes. Nevertheless, such method was first applied in a case study of CESAR.edu Master's Program Class and is being continuously used and improved in order to support the students learning process, providing quantitative results so that they can evaluate and progress their performance along the course. Software Engineering is the core discipline that guides the program, and the implementation of Software Factories is the mean to provide the Problem Based structure in the context of a masters program.

Key words: Method Evaluation, Problem Based Learning, Software Engineering, Software Factories.

1. Introduction

The ICT industry is emerging as an exciting and challenging environment – flexible and dynamic. These characteristics are demanded for ICT professionals, particularly software engineers, who should not only understand the technology, but also a wide vision of the problems, business understanding, entrepreneurship background and interpersonal skills, which are related to the practical experience of software engineering, according to Santos et al (2007).

Within this context, the term "Software Engineering" was originated in 1965 but first come into currency in 1967 when study group on Computer Science of the NATO Science Committee called for an international conference on the subject. As Brian Randell and Peter Naur pointed out in the introduction to their edition of the proceedings, "The phrase ‘software engineering‘ was deliberately chosen as being provocative, in implying the need for software manufacture to be based on the types of theoretical foundations and practical disciplines, that are traditional in the established branches of engineering.", supported by the work of Mahoney (1990). This sentence opens several areas of potential disagreement. Just what are the "types of theoretical foundations and practical disciplines that are traditional in the established branches of engineering"? What would their counterparts look like for software engineering? What role does engineering play in manufacture? Could one assign such a role to software engineering? Can software be manufactured? Those questions had no definitive answers in the conference proceedings and among the future Software Engineers.

Studies about the goals of higher education have consistently linked them to the students' future professional careers (Tynälä, 1999), integrating theory with practice into the curriculum, promoting the students' acquisition of specific and general knowledge, both applied to solve real problems. In this context, the PBL (Problem Based Learning) method (Savery et al, 1995) has been applied in a master course in software engineering to improve the effectiveness of learning, promoting the ability of students to work together to solve problems. The students are divided into working groups or software factories, and every factory is supposed to resolve a real problem of a client.

PBL in Software Engineering is seen as a methodology that builds real software factories. In this way, students are exposed to real, team oriented, and distributed development organization staffed and managed by themselves under the guidance of faculty. Several students are professional developers, certified programmers and work in industry, too. These courses are hands-on courses that require student participation in one of the factories defined and due to a complex organization of factories and process, a detailed method of evaluation is required.

This work discusses the evaluation method of real software factories composed by students and, as a result of a real life experiment, also points to a number of lessons learned, which can very likely be replicated within similar contexts.

Therefore, the objective of the discussion presented in this work is to analyze traditional evaluation methods available on the literature and present the method evaluation defined and its application, in the context of a software engineering master. In addition, it discusses the correlation that such process makes with the traditional evaluation methods and analyzes the results achieved in the application of such process in a class of students.

This paper is organized as it follows: Section 2 presents a bibliographical review on evaluation principals and concepts, followed by the explanation of the Master's Course Program structure on Section 3. Section 4 describes the evaluation approach defined for this context, Section 5 presents the mapping between the conceptual approach and its application to the program reality and Section 6 presents the case studies results. Finally, the concluding remarks and future works are presented on Section 7.

2. Assessment and Evaluation Principles and Concepts

The Problem-Based Learning – PBL (Savery et al, 1995) is frequently a difficult culture for new students, due to its characteristic of being an unusual learning paradigm. The educators also face difficulties emerging from the use of PBL techniques in their classes. Besides the critical challenge of finding and dealing with a real problem, educators have to deal with the decision on how to evaluate the technique's effectiveness and how to appraise whether students have met the overall learning objectives for the course.

The traditional evaluation methods based solely on individual examinations have little contributions in truly assessing a student's understanding and their practical experiences. In fact, students are well tested on individual examinations for discipline-specific content knowledge, but the group process skills needs to be improved, focusing on the areas of collaboration, communication, and critical reasoning to solve problems. The core PBL method is mainly concerned in assessing the practical area of the learning experience.

To define an effective evaluation model, in the context of a master's course program in software engineering, three aspects may be considered and analyzed: i) the principles to useful assessment (Waters & McCracken, 1997); ii) the different kinds of evaluation (Westat, 2002) and the profile of adult learners, which are detailed in the following Sections.

2.1 The Principles to Useful Assessment

- In (Waters & McCracken, 1997), three guiding assessment principles are indentified:
- **Content**: the assessment may reflect which are the most important topics for students to learn;
- **Learning**: the assessment enhances the learning and supports the instructional practice and;
- **Equity**: the assessment supports the learning opportunity for every student.

The PBL method provides a multidisciplinary approach for the solution of problems. Particularly, on software development projects, where different technical skills in solving a progressive problem are needed, the PBL application follows a specific practice: it begins with a general description of a system requirement and is developed to a specific and detailed system architecture and implementation. In this context, to know how to apply a concept of a user's requirements discipline to solve a problem of a software factory allow the students to pass through the true understanding and critical reasoning rather than solely memorization. The concept principle accentuates that the assessments should never be trivialized for the convenience of judgment, but rather should emphasize problem solving, thinking and reasoning skills (Sharma, 2002).

The learning principle emphasizes that the assessment should be continued in the the learning process and may not be perceived as an isolated activity. In the software factory context, this means that it is necessary to manage and evaluate all software development processes, including artifacts production, product delivery, project schedule, project changes and planning, correcting and giving feedback to the development team along the software development. This methodology is focused on the constant monitoring and feedback of the software factories work, and the implementation of diverse kinds of evaluation in order to explore different aspects of the software development cycle.

With respect to the equity principle, the master course implements a well defined pedagogical methodology which includes individual and collective evaluation. The last one is focused on obtaining the overall result of each software factory and it is conducted by a monitor, with the roles of the monitoring and controlling the software factory works.

Finally, the in the proposed approach, the assessment and evaluation process are considered as part of the learning process.

2.2 The different kinds of evaluation

The National Science Foundation – NSF (Westat, 2002) outlines two main types of evaluation: formative and summative. The purpose of a formative evaluation is to assess initial and ongoing project activities and the summative evaluation assesses the quality and impact of a fully implemented project. In the others words, as said in the handbook (Westat, 2002), "When the cook tastes the soup, that's formative; When the guest taste the soup, that's summative".

This article proposes a combination of the two main methods of evaluation: formative and summative. On the formative perspective, the purpose is to execute a "process evaluation", investigating whether the project is being conducted as planned, assuring that the program and its components are beeing executed and, if they are operating according to the proposed plan. In this step, the evaluation is an early check done by the project staff, or the internal evaluator. A progressive evaluation is also implemented with the responsibility to collect information to determine the impact of the activities and strategies on participants, analyzing the alignment among members of each team and their individual performance from the perspectives of the members of the team including his/her own viewpoint.

On the summative perspective, information about outcomes and related processes, strategies, and activities is collected. This evaluation is checked by an external monitor. In the proposed approach, these tasks are executed by the project client or any other relevant stakeholder of the software factories. Usually this type of evaluation is needed for decision making, which may result on a roadmap definition of the project, leading to the status: "project can be continued", "modify and try again" and "discontinue the project", depending of the result of evaluation. This kind of evaluation reflects the market judgment, very usual in real world projects, where the client is focused on practical and clear results for his/her business. Therefore, it is fundament for the entire assessment and evaluation process.

2.3 Profile of adult learners

Considering the human aspects of learning, in some important tracks of how to lead the evaluation are commented in the context of adult's learning, which is totally aligned with the PBL approach. First, if the adults have a more significant learning when they are inserted in real life situations and its consequent experiences, then the evaluation could be a link between the contents and the real world situations. Second, if the adults are motivated to advance in the learning with rapid answers about their performance, the evaluation could provide quick feedbacks for the student. And finally, if the adult learning is more significant when he/she feels as part of a process, then the evaluation will be able to assist with activities in group, socialization of team members, among others.

3. Master Course Structure

The master program was conceived to make use of the Problem Based Learning (Waters, 1997) methodology, in the context of Software Engineering discipline. Such discipline is the focus of the master's course where the real application of PBL can be achieved through the implementation of software factories: taking real problems faced by today's software industry, guiding teams to implement software factories to support the solution of such problems.

Within this context, the program is organized in eight disciplines, and each one is divided into three modules: Basic, Intermediate, and Professional. The disciplines are structured with the content to support the software factories problem resolution, and they are based on the areas of SWEBOK (SWEBOK, 2004), which is the reference to define the disciplines of the course. The disciplines of the master course are:

- Software Factories;
- Application and User Interface Requirements;
- Project Management;
- Architecture Oriented Development;
- Reuse Engineer;
- Technology to System Interoperability;
- Verification, Validation and Deployment of Systems.

The overall program was designed to display weekly modules of each disciplined, where there are 15 hours of classes, 12 hours of guided practices and an exam by the end of the weak. The hours of practices are used to help students implement the theory, presented during classed, on their real problem faced by the software factories and, therefore, there are activity monitors to give support on the execution of the activities.

Within this context, the structure of the course is supported by the following roles:

- **Content Professor**: is responsible to select the content that will be present on a discipline and to chronologically organize its presentation. He is also responsible to guide the Tutor Professors on the material to be used during the classes.
- **Tutor Professor**: is responsible to teach the planned program, coordinate academic activities, act as a consultant in the specified discipline context, systematically asses students, and to provide, for the Software Factory Monitors, information about the practices required during disciplines.
- **Advisor**: is responsible to provide orientation and support on the students' scientific and academic activities and systematically assess and evaluate the work being developed.
- **Activity Monitor**: is responsible to track the execution of the discipline academic activities, guiding the execution of any activities demanded by the **Tutor Professor**, and maintain the discipline site updated.
- **Software Factory Monitor**: is responsible to track the development of software factories, by conducting the process of collective evaluation, which will be described in details on the Section 4.
- **Client**: demands the projects to the software factories, whose main activity is to categorize the demand in terms of a RFP – Request for Proposal. The client must be available to interact and track software factories according to the project being develop. The client is also responsible to answer the "client satisfaction evaluation", which is part of the evaluation method of the factories.
- **Students**: executes all activities.

The program is organized into 12 moths, and its structure is dived into two stages: **software engineering practice** and **applied research**. The first stage is when the students are required to set up a software factory, and in order to realize this, they are organized in groups. Each factory is responsible to solve a real problem, usually presented by the industry and partners, where they formally present a real necessity. The course monitors, then, forms a pool of projects and each software factory deliver a request for proposal. Each project, submitted by the clients, must be compliant to a set of requirements, according to what is described bellow:

1. Innovation;
2. Relevance to industry;
3. Applied business model; and
4. Process required for its development.

The second stage of applied research is when students work individually to evolve a practical study as part of the master's degree dissertation, guided by a Tutor Professor. During this process, students are required to follow a tight schedule in order to prepare his/her final work.

Therefore, the structure defined on this section is used as a basis to implement the master course evaluation method, which will be described in details on the next Section.

4. An Evaluation Method Proposal

The master's program evaluation was conceived to contemplate both practice and theory disciplines, but giving a greater weight on practical activities developed by the group. The objectives of the applied evaluation method, which was conceived based on the conceptual approach of PBL (Waters, 1997), described on Section 2, are:

- To provide a diagnostic tool to ensure students are progressing adequately towards achieving the desired learning goals;

- To provide a multidisciplinary approach to collectively assess students;
- To set up an assessment process as part of the learning practice;
- To create an assessment process that can be repeatable over time which independent on the context and given the same variables, they should produce equivalent results; and
- To provide students a feedback of their evolution in a fair and objective manner.

Based on these characteristics, the master’s program evaluation provides a balance between theory and practice, where the theory corresponds to the traditional evaluation methods of individually executing exams to students and the practical part brings a solid living practice to the reality of a class by the development of real problem based projects.

The theory part corresponds to **40% of the final average**:

- Each discipline module (basic, intermediate and professional) generates an exam;
- Each discipline exam grade must be greater than or equal to 5 (five).
- The final discipline average must be greater than or equal to 7 (seven);
- By the end of the semester, the exams from each module are summed up and dived by 3 in order to generate the **discipline individual average**;
- This grade is individually applied to each student.

The collective part corresponds to **60% of the final average**:

- Each kind of evaluation has a specific grade;
- By the end of the semester, all grades are summed up to generate the final collective average;
- This grade is collectively applied to all members of the software factory.
- This grade has 60% of the weight replicated in all seven disciplines individual grade.

The grading calculation can be summarized in the following equation:

$$\left(\frac{\sum_{i=1}^{ModuleQuantity} IndividualModuleGrade_i}{ModuleQuantity}\right) * 0,4 + \left(\frac{\sum_{i=1}^{ActivitesQuantity} CollectiveEvaluation_i}{ActivitesQuantity}\right) * 0,6$$

Therefore, it is observed that the student will only know his final average after the end of all activities. Figure 1 represents the correlation of theses evaluation parts, and each detailed evaluation will be described next.

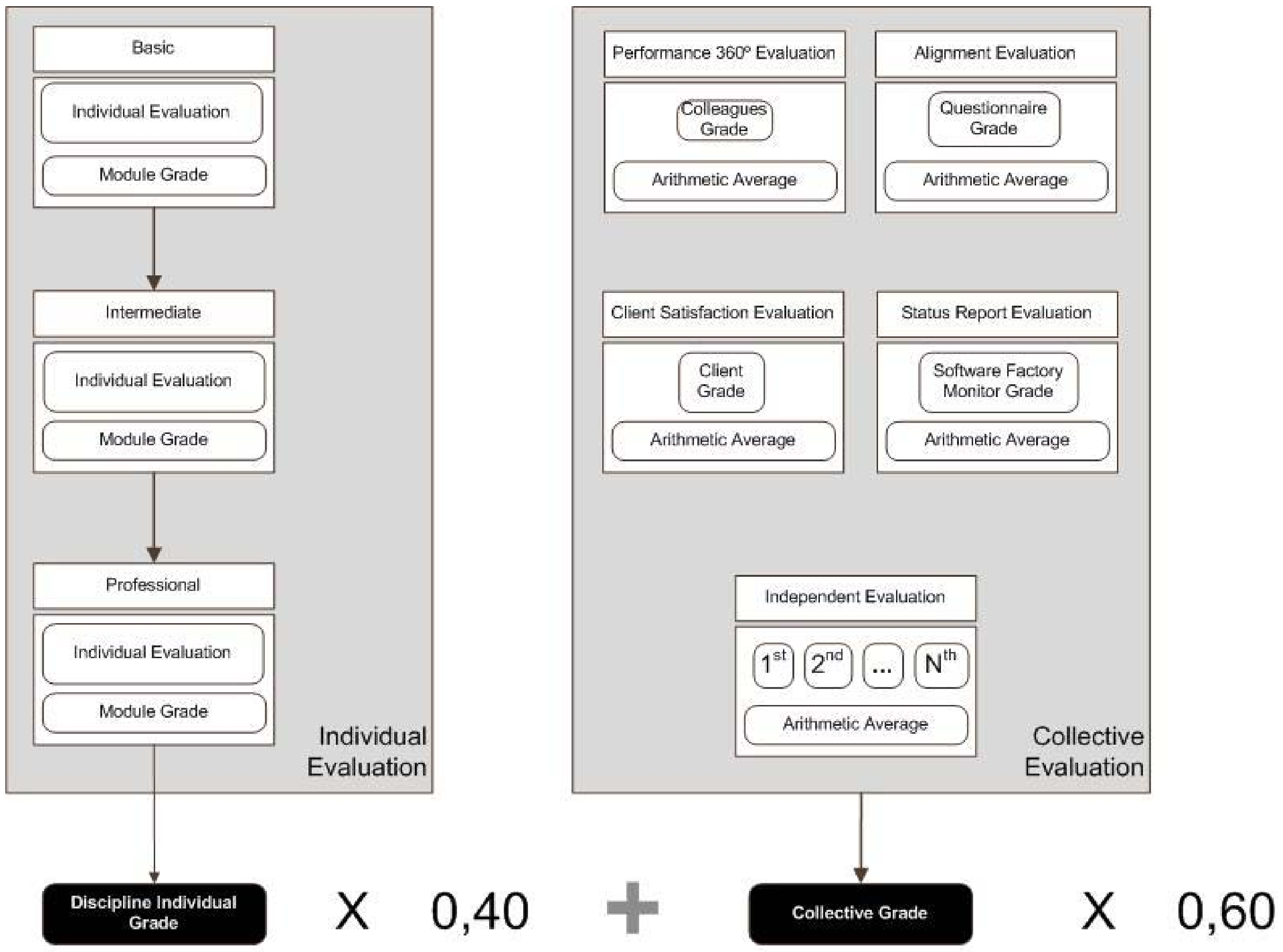


Figure 1: Evaluation Method Representation

4.1 Theory Evaluation

This evaluation focuses on the disciplines, where each module can have a configuration. The individual grade of each module is composed of an **Individual Evaluation**, which are evaluations realized by the end of each module, with multiple questions, or subjective dissertation elaborated with the support of Moodle[1].

4.2 Collective Evaluation

The grounding for the collective evaluation is formed by practical activities executed by the software factories teams. The abstraction levels depicted on Figure 2 represent the different models and data sources, from the bottom of the pyramid, that are gathered to run a software development project in the context of a software factories. These levels of abstraction are intended to be covered by the coordinated approach of evaluation, through looking across the deliverables of the bottom of the pyramid, examining the consistency among them.

[1] Moodle: A free Course Management System – CMS – developed as open source software to help educators create effective online learning communities
Available at http://www.moodle.org

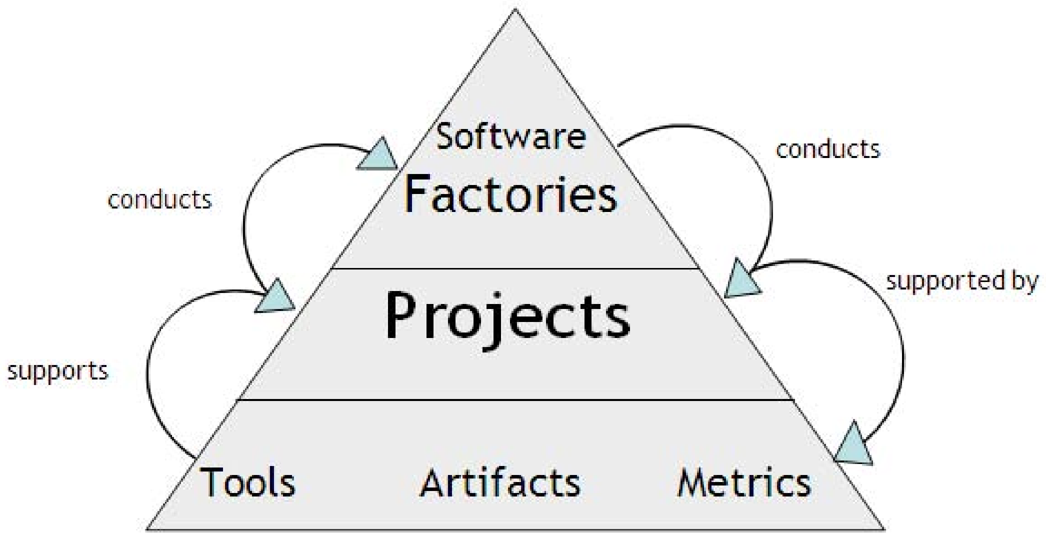


Figure 2: Evaluation Abstraction Levels

This evaluation can be made through one of the following resources:

1) **Independent Software Factory Evaluation**: in this moment, the factory monitor collects all the evidences and information from the factory and the project being developed with the objective to analyze the progress and continuity of the activities. In this moment, it is analyzed whether the artifact was delivered on the time agreed and the technical content of the product. For this evaluation, a product checklist is used to guide the grades given by each artifact and such checklist was elaborated based on the process areas of CMMI-DEV (2006).

2) **Alignment Evaluation**: this activity has to objective to analyzed where or not all team members, in the same factory, are aligned with the objectives and activities being developed. The evaluation, then, consists of a questionnaire that is distributed to all members of the same team and collection of the results. The questions are focused on the development of the factory and business, and the results are given considering the answers, whether they convert to the same view.

3) **Client Satisfaction Evaluation**: it is a set of questions sent to the client that forms a grade, from 1 to 10, and the client evaluates the team involvement. The result of the process is a grade given by the client.
4) **Status Report Evaluation**: this is the moment when the team realizes a formal presentation of the project and the process being executed. The factory monitor analyzes the factory presentation, project presentation, milestones tracking, Strong and Weak points, metrics, business related to the factory and consolidates the results achieved so far.

4.4 Concluding Remarks

It is well understood the complex structure presented by the evaluation method on the literature and, therefore, such methodology is applied as a guideline to be followed in order to achieve the desirable levels of acceptance, for both the faculty and the students. The method was defined taking the literature approach, and the mapping between the method and the theory will be present on the next Section.

5. Mapping the Evaluation Approach with the Educational Theory

The approach based on individual and collective evaluation with more emphasis on the collective part is justified by the characteristics of PBL method and its emphasis on practical experiences through the work in group. In this kind of methodology, questions as the instructional components involved in the evaluation process (content, learning and equity principles), how the evaluation process is applied (formative and summative) and for one (post-graduation students) need to be defined and systematized, allowing the definition of control’s points and performance metrics for learning’s effectiveness validation.

On the people's point of view, the learning process is entirely multidirectional in PBL. Students receive guidance and support from his/her friends and peers, besides the tutor and monitor. Specifically in a master course, the students are actual professionals in the ICT industry, therefore they have some maturity level and prior experience of work that influences the learning process. These influences demand for pedagogical components associated with the profile of adult learners mentioned in the Section 2.3, such as practices based on situations of the real life, quick feedbacks and activities in group. In this context, the practical experiences and professional maturity of tutors and monitor are essential.

Analyzing the principles for useful assessment and the formative and summative kind of evaluation, it´s possible to do an association with the components of the evaluation approach proposed in this article, as showed in the Table 1.

	Instructional Components		
kind of evaluation	**CONTENT**	**LEARNING**	**EQUITY**
FORMATIVE	Independent evaluation	Status report	Performance in 360o. evaluation
SUMMATIVE	Modules evaluation (basic, intermediary and professional)	Satisfaction client evaluation	Alignment's questionnaires

Table 1*: Association between evaluation approach and educational theory.*

Here, the combination of kinds of evaluation and principles proposes more control in the software factories towards quality of its results (software and process) and, consequently, quality of educational objectives.

On the formative perspective, the independent evaluation is focus on the artifacts and partial products generated in the software factories with the support and consulting of the discipline's tutors. This evaluation is conducted by software factory monitor, who revised project's documents as project plan, user's requirements specification, project schedules, quality plan, among others, and assure the publication of this information for the client.

About the learning component, the implementation of status report allows investigating whether the project is being conducted as planned, make sure the program and its components are really operating according to the proposed project plan. It is important to emphasize that this plan was defined by each team in accordance with its client and approved by software factory monitor. Therefore, any change or adaptation of this plan needs to be negotiated with the respective stakeholders.

The formative evaluation also include a performance evaluation, step in which each team member evaluate himself/herself and the others members of his/her team. Questions about leader behavior, participation, communication skills and commitment of the participants are made with the objective of investigating the motivation and professional maturity of students, offering the opportunity to improve their performance.

On the summative perspective, individual module evaluation is focused on the discipline-specific content evaluation by the discipline tutor, but not isolated of the practices in the software factories. In this step, the content application is evaluated totally contextualized in the specific project that each software factory is developing. One example of such questions is "why or why not the framework defined by Greenfield could be applied in your software factory?". The presence of subjective questions is essential, which tests no solely the understanding of the Greenfield´s framework but its applicability and convenience for a real problem to solve. Additionally, an alignment evaluation is ran with the objective of verifying the knowledge uniformity among the team members with respect to process and products produced. Finally, each client evaluates the software factory results through satisfaction criteria questionnaires, at least in two moments in the project development.

6. Case Study

6.1. Scenario and Objectives

The case study was conducted with the main objective to assess the implementation of the evaluation method in the class of CESAR.edu master's program. The class was composed 19 students, with 26,84 of age average and they were professional whose abilities were in the most of the cases: test engineer and quality engineer. This assessment happened from August 2007 to February 2008, during the first stage of the master's program of this class.

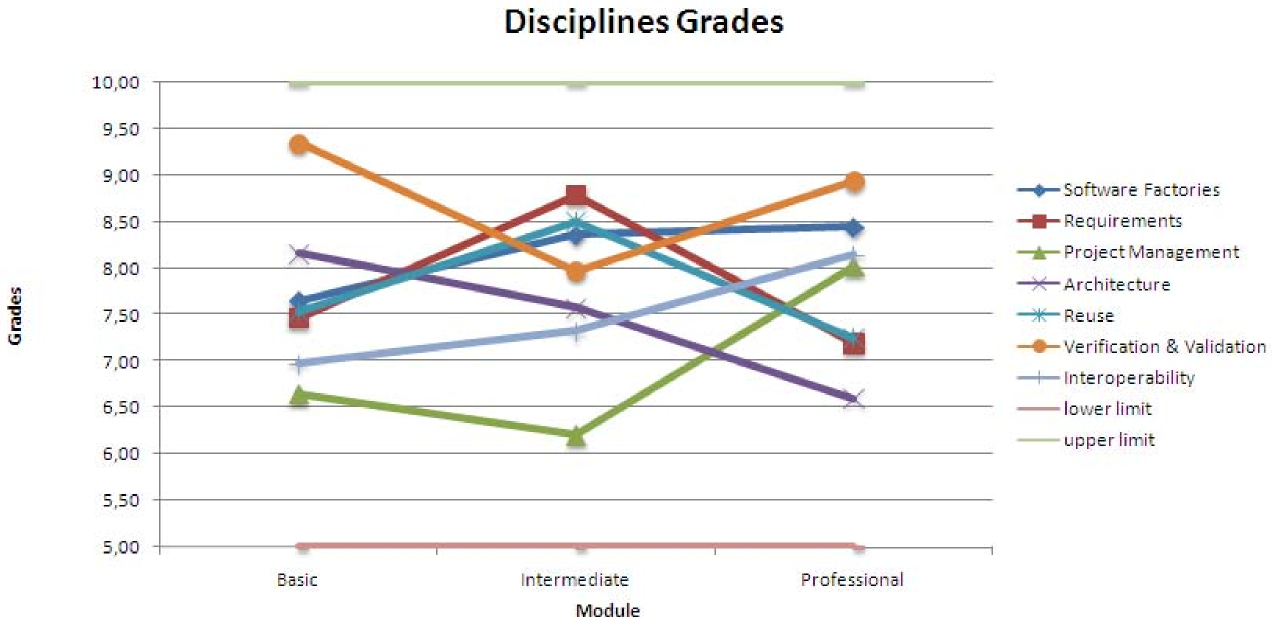
This class was the first one formed on CESAR.edu program, and therefore, the first time that the evaluation approach was being executed and needed to be improved for the following classes. Accordingly, by the executing the methodology, the master program general goals could also be analyzed to indicate whether or the class would be improving their technical abilities as a result of the support and knowledge aggregated by the course.

The results will be assessed in group, not isolated cases because the intention is to verify whether or not the class is having progress.

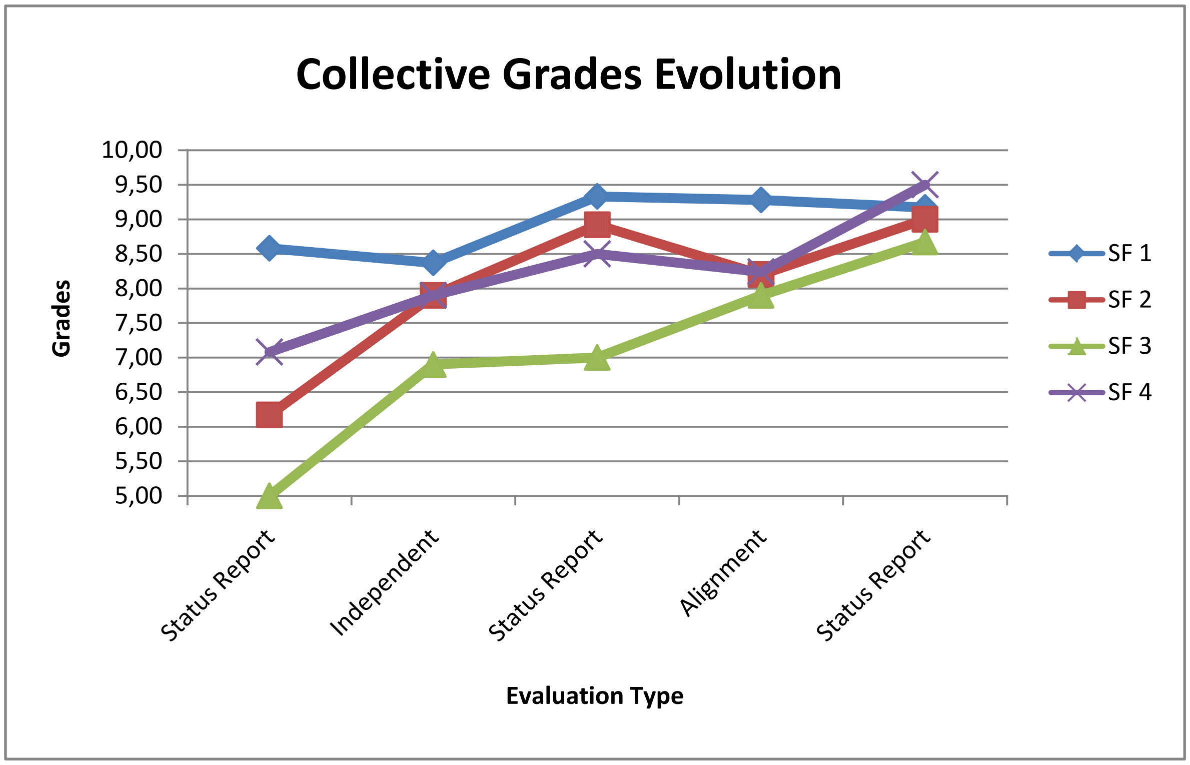
6.2. Collected Data.

Based on this scenario, the results of the two types of evaluation were collected and will be presented bellow.

This first graph represent an average of the discipline grades, from each module, achieve by the student. It is seen that, based on the average, all disciplines modules grades were greater than 5 (the minimum grade expected). Three, out of seven disciplines (Software Factory, Project Management and Interoperability) presented an evolution when comparing the grade from the basic module with the grade from the professional module, where the last one was greater than the first one. This corresponds to 43% of the disciplines presented a progress based on the theory part of the evaluation method. This behavior is expected once the professional modules of each discipline are much more complex when related to the basic modules, and, consequently, requires a greater maturity of the students. In the software factory discipline the performance of the students is good, and it is due to the work of the software factory monitor in continuously assessing and observing the students and their projects. The performance of the students in the disciplines Project Management and Verification & Validation are good due to the professional profile of most of the students – Test engineer and Quality engineer.



In this second graph, the evolution of the collective grading is presented, based on the execution of six evaluations (3 Status Reports, 2 Independent and 1 Alignment) in each one of the four software factories. By observing the lines of the graphics, it is seen that three out of four software factories presented a final grade, on the last status report, greater than the first independent grade. The only software factory that presented a final grade smaller than the first grade presented a behavior that was ahead of the other factories during all other evaluations and his collective final grade was higher than the others. It is also observed that the final grade of the status report converge to a range that varies in only 0,83 points (the higher grade was 9,50 and the lower grade was 8,63). The biggest progress observed is from software factory 3, where the difference from the first grade and the last grade is 3,67 points, representing a progress of 73%.



7. Conclusion and Future Work

The deployment of evaluation process into the context of the master program, during the process of improving the maturity of such methodology, has presented considerable results indicating positive remarks to the software factories organized by the teams. By implementing theory and practical evaluation, an explicit improvement could be observed, and consequently brought us a feedback about the program definition and its effectiveness on group of students.

On the other hand, it was also important to implement the students' feedback to stabilize and improve the evaluation method in order to maintain a continuous improvement in the evaluation methodology. The goal of implementing a method to appraise the results delivered by some procedures, within projects, guided by processes and products, could only be measured due to the designed methodology and an ad-hoc manner could realize misjudgment of the acquired results.

Nevertheless, the innovation of the approach applied is recognized as something that can bring great benefits to all members involved, but in order to achieve that, caution is strictly needed to generate the most accurate grading scales to the evaluation.

Besides these considerations, some future work and improvement opportunities can be seen for this process:

- The elaboration of checklist to best guide the individual evaluation of artifacts and to guide them on their elaboration;
- The elaboration of checklist to best guide the software factory process evaluation;
- The schedules of delivery required by the artifacts on each discipline should be synchronized with the evaluations made by the Software Factory Monitor; and
- The tutor professors, who are specialist on each discipline, should support the Software Factory Monitor in evaluating the artifacts of their discipline, so that the technical content could be best analyzed.

About authors:

M.Sc. Ana Paula Carvalho Cavalcanti: is a research professor of the Educational Unit of C.E.S.A.R.edu and Quality Engineer of the development projects from C.E.S.A.R. Received her Bachelor degree from State University of Ceará – UECE (Brazil) in 2005 and Master degree from the Informatics Center of the Federal University of Pernambuco – UFPE (Brazil) in 2008. With over 5 years of experience in software development, her major research interests focus on software quality, software reuse processes and software development education.

Ph.D. Simone Santos is a coordinator of the Master course in software engineering of the Educational Unit of C.E.S.A.R.edu. She has been working at C.E.S.A.R since 1997, assuming diverse roles such as e-Business Unit leader, Products Unit manager and innovative software projects manager. Today, she is technical leader in

C.E.S.A.R.edu coordinating the faculty and and also advising students their master's research work.

Ph.D. Maria da Conceição Moraes Batista received her Bachelor degree, Master degree and PhD in computer science from Informatics Center of the Federal University of Pernambuco – UFPE (Brazil) in 1988, 2003, and 2008 respectively. Currently she is a regular professor of the Educational Unit of C.E.S.A.R.edu with over 10 years of software development. Her major research areas are information quality, databases, Web systems and software development education.

Ph.D. Jones Albuquerque has achieved his Bachelor degree and Master degree from Informatics Center of the Federal University of Pernambuco – UFPE (Brazil) in 1994 and 19997 and concluded his Ph.D in computer science from the Federal University of Minas Gerais – UFMG (2002). Nowadays he is a current professor of the Rural Federal University of Pernambuco – UFPR and CESAR collaborator. Has experience and interest in the research areas of Computer Science, Software Engineer, Mathematics Modeling, Software Process, Software Quality, Computational Epidemiology and System Modeling.

Ph.D. Silvio Meira has 23 years of experience in advising and execution of scientific, technological and innovation research, and created the Ph.D program from the Informatics Center of the Federal University of Pernambuco – UFPE (Brazil) and coordinated the post-graduate program during 7 years. Besides that, he plays an important role at C.E.S.A.R.edu as a chief scientist, and recently has been considered by Info Exam Magazine as on of the 100 most important people on information technology in Brazil.

References

CMMI-DEV (2006), CMMI for Development, V1.2 model, CMU/SEI-2006-TR-008. Software Engineering Institute.

Mahoney, M. S. (1990). *The roots of software engineering*. Technical Report CWI Quarterly 3-4, Princeton University, 325-334.

Santos, S. C., Albuquerque, J. O., & Meira. S. R. L. (2007). *Integrating Concepts and Practices: PBL Applied to a Master Course in Software Engineering*. – White paper.

Sharma, S., Sugumaran, V., & Rajagopalan B. (2002). *A framework for creating hybrid-open source software communities*. Info Systems, 12:7-25.

Savery, J.R., Duffy, & T.M. (1995). *Problem based learning: An instructional model and its constructivist framework*, Educ Technology.; 35(5):31-7.

SWEBOK (2004). *Software Engineering Body of Knowledge*. IEEE Computer Society.

Tynälä, P. (1999). *Towards expert knowledge? A comparison between a constructivist and a traditional learning environment in the university*, Int. J. Educ. Res., v.31, p.357-442.

Waters, R., & McCracken, M. (1997). Assessment and Evaluation in Problem-Based Learning.

Westat, J. F., (2002). *The 2002 User Friendly Handbook for Project Evaluation*. The National Science Foundation, Contract REC 99-12175.