

# A Qualitative Study on PATT – A Project Assessment and Tracking Tool

Fabio Perez Marzullo

Federal University of Rio de Janeiro – UFRJ,  
COPPE Software Engineering Laboratory, Brazil  
fabio@mz-empresarial.com.br

Geraldo Bonorino Xexéo

Federal University of Rio de Janeiro – UFRJ,  
COPPE Database Laboratory and DCC/IM, Brazil  
xexeo@cos.ufrj.br

**Abstract—** This paper describes PATT, a project assessment and tracking tool, created with the purpose of processing software metrics in order to provide managers with development information on software projects. Based on international software standards, including CMMI, ISO/IEC 9126 and ISO/IEC 15939, it collects metrics from projects artifacts, analyzes them, and spend efforts on calculating overall development progress. Designed to be the main component of an analysis framework, PATT implements a set of functionalities capable of executing such analysis seeking knowledge over undergoing software development.

**Index Terms—** Metrics, Product Quality, Quality Programs, Quality Control Tools and Methods.

## I. INTRODUCTION

Researches conducted along recent decades have shown that measurement activities are essential tools in software project management [1]. Measurement is important to software development, testing and maintenance; and, to obtain a successful measurement process, it is necessary to seek guidance among standards such as ISO/IEC 9126 [2] and ISO/IEC 15939 [3]. Measurement is now considered basic software engineering practice, as evidenced by CMMI's Measurement and Analysis Process Area [4].

The way measurement processes are actually implemented and used, in software organizations, might

This work was supported in part by the Federal University of Rio de Janeiro – UFRJ, COPPE Software Engineering Laboratory and Marzullo Soluções em Gestão Empresarial Ltda. A Qualitative Study on PATT – A Project Assessment and Tracking Tool.

Fabio Perez Marzullo, M. Sc. is with Federal University of Rio de Janeiro and is Marzullo Soluções em Gestão Empresarial Ltda CEO, Rio de Janeiro, Brazil. (phone: 55-21-3975-1565; e-mail: fabio@mz-empresarial.com.br).

Geraldo Bonorino Xexéo, D. Sc. is with Federal University of Rio de Janeiro and COPPE Database Laboratory, Rio de Janeiro, Brazil, (e-mail: xexeo@cos.ufrj.br).

determine whether they will succeed or fail the development process [5].

The *Practical Software Management (PSM)* [6] states, “Top-performing organizations design their technical and management processes to make use of objective measurement data. Such data and associated analysis results support both short and long-term decision-making”. The PSM defines a Measurement Process, which describes purposes and outcomes of a compliant measurement processes, along with associated measurement activities and tasks.

Measurement Activities play an important role when development information needs to be acquired from software projects. However, what we find in many companies is the absence of measurement programs<sup>1</sup>. Although it is recognized as an essential management tool, the reluctance in conducting such programs has a significant source: implementing it in an adequate and successful manner is a challenging task [4].

That happens not only because most managers lack the necessary knowledge, but also because of the overload of existent metrics. Moreover, implementing measurement programs needs the commitment of everyone involved in the project, as well as assistance of automated tools.

A survey conducted by the Brazilian Program of Software Quality and Productivity (PBQP) [7] from 1995 to 2003, has shown that efforts regarding improvements on software development management increases everyday. However, the actual number of quality control programs running in the market today is still very low. In 2003, for example, 35% of inquired companies conducted quality programs along software development; 20% knew and implemented CMMI activities and only 19% used function points technique to assess software quality. At that time, 3.5% of

<sup>1</sup> Different assessments estimate from 38% to 70% of IT companies at CMM level 1, what is a strong indicator of the lack of measurement activities

companies annual investments were applied to employee certifications programs.

Therefore, the survey showed that most companies did not use metrics to assess software development. In addition, the lack of automated tools contributes in lowering the number of companies engaged in some sort of Management Program.

Thus, two questions are set in the discussion:

- How can a company conduct a Management program in an easy and successful manner? and;
- How can the program be automated and, hence, lower overall complexity?

The best approach to initiate a measurement program is to seek guidance in software standards [1,2,4,8,9,10,11]. In addition, we point to the fact that the use of automated tools strongly helps to decrease the difficulties regarding such implementation.

Tools have the ability of simplifying complex tasks and generate quicker analysis results [12]. Furthermore, availability of the right information, on time, improves software process quality, and has a positive effect in software products.

## II. DEFINITIONS

PATT stands for *Project Assessment and Tracking Tool*. It is part of a software analysis platform capable of manipulating object-oriented metrics. The tool's purpose is to acquire measures from different software projects artifacts and compare them, trying to establish overall software development progress. The example below explains the idea.

Suppose we want to verify whether requirement R1 is implemented, or what is its current development status. During software development, two relevant artifacts are constructed: (1) a model diagram and (2) a source code (including packages and classes). PATT acquires measures from both artifacts, transforms them into PATT's own representation (a metadata model specifically designed to cut down unnecessary complexity) and then proceeds with the analysis, by comparing model diagram measures with source code measures. By doing this, PATT checks whether both measures set matches, which means that, it calculates whether a class C1, defined in the model diagram, is closely implemented in the source code packages. If PATT finds class C1 in any source code package it then identify if its attributes and methods are present, compare both measures and analyses test results applied for that specific class. After concluding the analysis, it presents a report summarizing the actual development progress of every class created in the software.

## III. PATT FRAMEWORK

The framework defines PATT's analysis architecture. When modeling the framework, it was necessary to establish a measurement process, composed by measurement activities, in order to create a standard and repeatable measurement program.

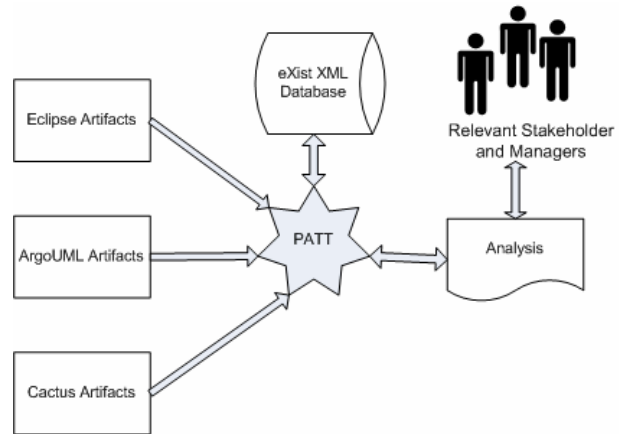


Figure 1. The figure describes basic aspects of PATT's framework. Artifacts are assessed by PATT and analysis results are available to relevant Stakeholders and Managers

## IV. PATT PROCESS MODEL

PATT Process Model conception aimed creating a process model capable of adhering to any development processes [24, 25]. The process activities used in PATT's framework are described below:

1. *Business Modeling & Requirements Elicitation*. This phase is responsible for modeling business processes. It produces requirements documents, which represent software construction needs.

2. *Analysis & Design*. This phase is responsible for developing business and application models. These models specify characteristics; experiment architecture designs and creates software logical view.

3. *Software Implementation*. This phase is responsible for coding the software. It creates physical software implementation (mostly source packages and classes).

4. *Software Testing*. This phase guarantees minimal software compliance to requirements and quality.

## V. PATT WORKFLOW

The workflow is responsible for executing software analysis, by executing measurement activities defined in the process model. The Workflow activities are:

1. Produce and select relevant artifacts;
2. Determine relevant metrics;
3. Acquire measures regarding selected artifacts;
4. Execute Analysis;
5. Store Measures;
6. Handle communication issues.

Figure 2 describes the assessment workflow. It presents how PATT's Process Model and PATT's Workflow works inside PATT's Framework.

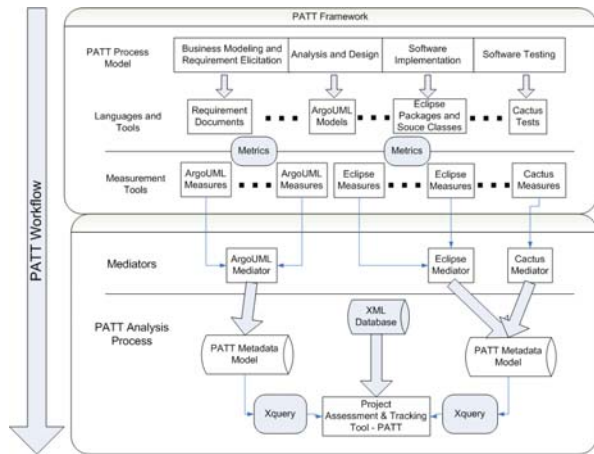


Figure 2. PATT's Framework: software development model and measurement activities

## VI. PATT'S ACTIVITIES EXPLAINED

### A. Select Relevant Metrics

This key activity is responsible for eliciting projects information needs. When selecting metrics from the base metrics set, managers might focus only on those metrics that best represent information objectives.

### B. Acquire Measures Regarding Selected Artifacts

This activity is responsible for gathering XML documents generated by development tools. All XML documents must contain measures extracted from project artifacts. Once they are available, PATT's mediators must take over control and transform such specific XML documents into PATT's Metadata representation.

### C. Transform XML Documents into Metadata Model

Mediators transform generated XML documents in Metadata Model, in order to create a unique measurement representation. This unique representation standardizes artifact manipulation, which should allow PATT to assess project under different development paradigms. After performing the conversion, PATT executes a set of routines to load all measures inside its analysis environment.

### D. Analysis

PATT analysis activity is responsible for assessing software development progress. It establishes an environment where software metrics analysis produces relevant information, in order to gain project insight [28].

For example, suppose we want to verify whether a java source code Ji (including packages and classes) matches a model diagram Mk. To validate its current development status, PATT does the following actions:

1. First, convert source code XML document into metadata representation and next it imports all measures in that XML document. Figure 3.1 presents code information:

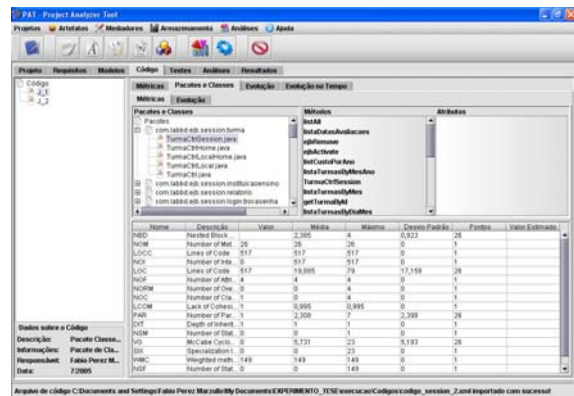


Figure 3.1. Code packages measures imported by code mediator.

2. Second, convert model diagram document into metadata representation and next imports all measures in the XML document. Figure 3.2 presents model information:

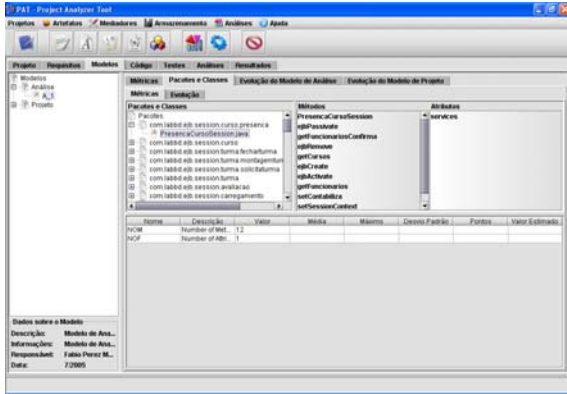


Figure 3.2. Model measures imported by model mediator.

- Third, it executes analysis procedures and compares model measures with source code measures, calculating whether classes described in Mk, are created in Ji. Figure 3.3 presents analysis results.

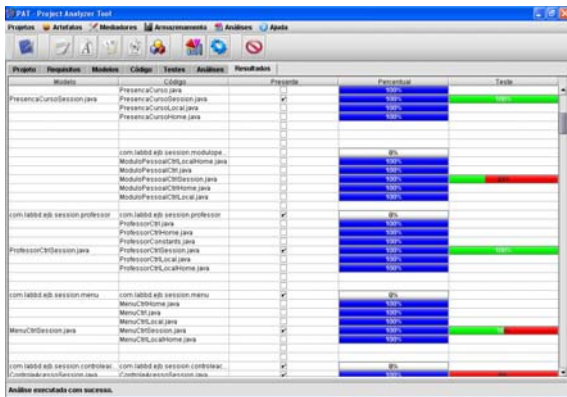


Figure 3.3. Analysis execution.

- Fourth, it generates graphics to report development evolution through time. Figure 3.4 presents such evaluation.



Figure 3.4. Results from analyzed data.

## E. Metadata Model

The process of creating Metadata Model representation has been influenced by the decision of allowing proper manipulation and storage of acquired measures. It has become clear that XML Documents generated by development tools would overwhelm PATT, since it would require it to manage too many different formats.

Benefits from this involve simpler storage procedures, analysis functions and overall data manipulation.

Managing measures in Metadata Model accelerates, enhances, and simplifies access to stored data; therefore, it improves data organization and management by focusing on the creation and manipulation of metadata model-formatted documents.

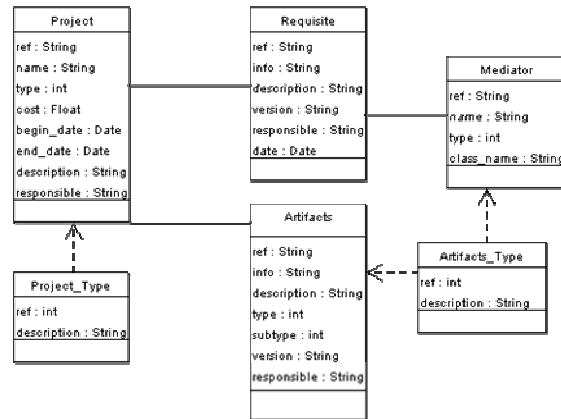


Figure 4. PATT's Metadata Model

## F. Mediators

Mediators are components responsible for transforming proprietary XML documents into Metadata Model XML documents. PATT was developed using Java Language, therefore, Mediators should also be developed in Java and implement a generic interface called *Mediator*. The *Mediator Interface* contains a single method named “*execute()*”, which is called by PATT when a mediator is requested to perform XML transformation. This rule enables third party Mediators to be added by PATT, allowing proper generalization and manipulation of different kinds of artifacts.

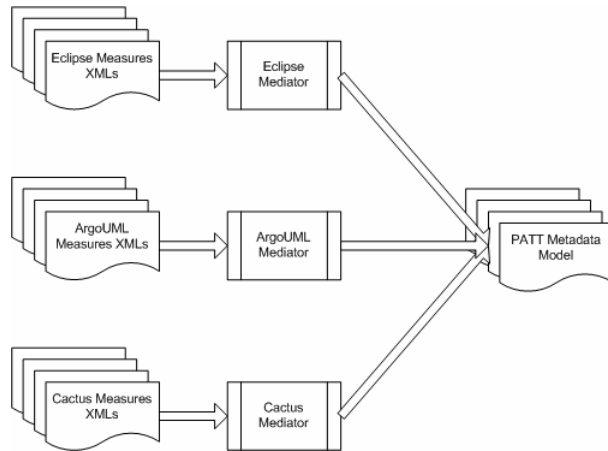


Figure 5. Measures XMLs must have their own mediators to allow PATT to execute necessary analysis.

### G. Communication

Communication is available from an internal interface that exposes analysis results using graphics and progress tables.

### H. Storage

PATT storage mechanism works with Native XML Databases [29]. In order to maintain reliable analysis data, we decided to store all XML Document in an XML Database. The use of XML databases helps PATT's scalability, since it manipulates generic XML information.

XQuery language [31,32] is the main component of PATT's storage procedures. It simplifies the way in which the database handles data. PATT's storage procedures are responsible for gathering all XML documents generated during analysis phase and storing them into the Database.

## VII. QUALITATIVE STUDY

The research validation focused on Java projects. We used a major Petroleum Brazilian Company web system to calculate, along different development periods, its development status. The observation towards all gathered data allowed a major insight regarding the system development behavior.

### A. Planning the Study

The assessed project had the objective of controlling the company's employee's professional certification

programs, as well as, providing necessary support for exams preparation and online tests. The project should conform to thirteen functionalities, listed below:

1. Access Control;
2. Exam Management;
3. Short-Term Course Management;
4. Certificate Printing;
5. Subject Management;
6. Long-Term Course Management;
7. Internationalization;
8. School Assessment;
9. Employee Management;
10. Profile Management;
11. Teachers Control;
12. Tests and Questions Control;
13. Managerial Reports;

All project artifacts were developed using known development programs, for example, analysis and design models were constructed using the ArgoUML [34] tool, source code classes and packages were developed using Eclipse [35], and all tests were constructed using the Cactus framework [38].

### B. Practitioners

The study was conducted by the projects manager and involved ten professional, combined into to groups:

1. Analysts: responsible for business rules acquisition and modeling the process;
2. Developers: responsible for coding the project.

### C. Study Proceedings

These activities have been conducted along the development process, in order to gather necessary information:

1. Artifacts Construction;
2. Measurement Acquisition (in XML Format);
3. Migration of Measurement XML files into Metadata XML Format;
4. Import Measures into PATT;
5. Analysis Execution;
6. Results Assessment.

### D. Study Results

The study was conducted in two phases. The first had the objective of estimating expected project results. The project manager estimated, for example, the amount of source code classes that would be necessary to construct

the software and inputted those values inside the tool. The second phase regarded the acquisition of actual values, by conducting the analysis activities. In possess of such information, the comparison (of both estimated and actual measures) showed that the project had a calculated mean delay of 14.63% of the number of modeled classes. It means, for example, that, when planning the software, the first model version should have fifty classes, however the actual value, measured at the established milestone was only forty-four classes. Table I presents the assessment conducted between analysis model estimated and actual values.

TABLE I  
ANALYSIS MODEL ASSESSMENT

Versions	Estimated	Actual	Delay
1	50	44	12.00 %
2	55	47	14.54 %
3	55	47	14.54 %
4	60	50	16.66 %
5	60	51	15.00 %
6	60	51	15.00 %

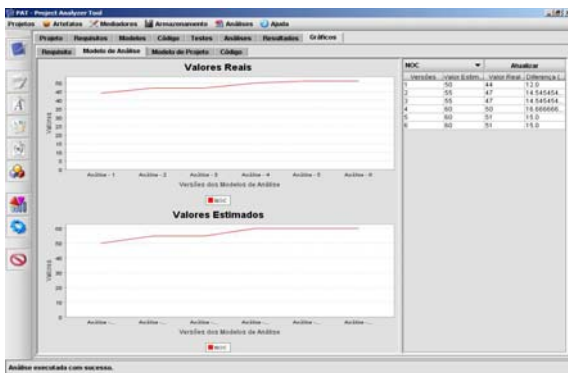


Figure 6-A – Analysis Model Assessment.

Regarding source code packages the overall delay accounted for only 5.35 %, according to the assessment presented in table II.

TABLE II  
SOURCE CODE ASSESSMENT

Versions	Estimated	Actual	Delay
1	50	45	10.00 %
2	60	56	6.66 %
3	60	56	6.66 %
4	65	59	9.23 %
5	65	61	6.15 %
6	65	62	4.61 %
7	70	67	4.28 %
8	80	79	1.25 %



Figure 6-B – Source Code Assessment.

Finally, requirement development evolution is shown in table III. It presents requirement development evolution according to source code construction.

TABLE III  
REQUIREMENT DEVELOPMENT VALUES

Requisites	Versions	Development
Access Control	1	100%
Exam Management	1	55%
ST Course Management	1	89%
Certificate Printing	1	11%
Subject Management	1	100%
LT Course Management	1	90%
Internationalization	1	22%
School Assessment	1	91%
Employee Management	1	78%
Profile Management	2	100%
Teachers Control	2	92%
Questions Control	2	81%
Managerial Reports	2	93%

## VIII. CONCLUSIONS

This paper presented PATT, a Project Assessment and Tracking Tool, defined in a low cost object oriented development framework. It provides the background for understanding PATT's Analysis process, applying object-oriented metrics, in order to calculate overall software development progress.

Along with the study, this paper presented relevant characteristics of the assessment process created to analyze software development. Following all defined rules, it is possible to analyze software development using software metrics.

The intention was to obtain development information in order to allow relevant project stakeholders to make proper decisions regarding software planning. According to the study results, PATT was able to expose flaws and delays during the project development schedule.

In general, it succeeded in proving that software assessment using scalar metrics is possible. Despite there are still many aspects that consider revising, PATT behaved as an important learning agent, collecting and exposing projects information.

#### REFERENCES

- [1]. Ishigaki, D.; Jones C., "Project Management in the Rational Unified Process". 29/10/2001. CS2 Software Engineering note 3.
- [2]. ISO/IEC 9126 - *Software Engineering - Product Quality*.
- [3]. ISO/IEC 15939 - *Software Engineering - Software Measurement Process*. 2002.
- [4]. SEI Software Engineering Institute, *CMMI - Capability Maturity Model Integration* 2002.
- [5]. Rozum, J. A., "Defining and Understanding Software Measurement Data", SEI Publication Visited in 03/02/2005.
- [6]. McGarry, J., Card, D., Jones, C., Layman, B., Clark, E., Dean, J., Hall, F., "Practical Software Measurement: Objective Information for Decision Makers". Addison Wesley Professional.
- [7]. Programa Brasileiro da Qualidade e Produtividade - PBQP. Indicadores e Metas da Qualidade e Produtividade em Software <http://www.mct.gov.br/Temas/info/Dsi/PBQP/Indic.htm>. 2001.
- [8]. Rocha, A. R. C., Maldonado, J. C., Weber, K. C., "Qualidade de Software - Teoria e Prática", Primeira ed. Rio de Janeiro: Prentice Hall, 2001.
- [9]. ISO/IEC 12207 - *Software Engineering - Life Cycle Processes*.
- [10]. Project Management Institute, "A Guide to the Project Management Book of Knowledge", 2004 Edition ed. Newtown Square, Pennsylvania USQ: 2000.
- [11]. Ogasawara, H., Yamada, A., Kojo, M., "Experiences of Software Quality Management Using Metrics through the Life-Cycle". 1996. IEEE Proceeding of ICSE-18
- [12]. XQuery 1.0: An XML Query Language. [www.w3.org/TR/xquery/](http://www.w3.org/TR/xquery/) 2004.
- [13]. Simmons, D. B., Ellis, N. C., Fulihara N., Kuo, W., "Software Measurement - A Visualization Toolkit", New Jersey: Hewlett-Packard, 1997, pp. 3-442.
- [14]. Royce, W., "Pragmatic Quality Metrics for Evolutionary Software Development Models". 1990.
- [15]. Basili, V., Phillips, T-Y, "Evaluating and Comparing Software Metrics in the Software Engineering Lab" 1981.
- [16]. Ben-Menachem, M., Gelbard, R., "Integrated IT Management Tool Kit," 2002.
- [17]. Zuse, H., "A Framework of Software Measurement". 1997.
- [18]. Sveiby, K. E., "What is Knowledge Management". 2001.
- [19]. Takeuchi, H., "Beyond Knowledge Management: Lessons from Japan". 1998.
- [20]. ISO/IEC 14598-5 *Information Technology - Software Product Evaluation*. [Primeira Edição]. 1-7-1998.
- [21]. Dutta, S., Wassenhove, L. N. V., Kulandaiswamy, S., "European Software Management Practices". 1998. Communications of the ACM.
- [22]. Dufour, B., Driesen K., Hendren L., Verbrugge C., "Dynamic Metrics for Java". 26/10/2003. ACM - OOPSLA 2003.
- [23]. Ishigaki, D., Jones, C., "Practical Measurement in the Rational Unified Process". 2003.
- [24]. Pressman, R. S., "Software Engineering - A Practitioner's Approach. Fifth Edition". 2003.
- [25]. Jaaksi, A., "Assessing Software Projects - Tool for Business Owners". 01/08/2003. ESEC/FSE. Nokia.
- [26]. Chidamber, S. R., Kemerer, C. F., "Towards a Metrics Suite for Object Oriented Design" 1991.
- [27]. Lorenz, M., Kidd, J., "Object-Oriented Software Metrics" 1995.
- [28]. Kafura, D., "A Survey of Software Metrics" 1985.
- [29]. Salminen, A., Tompa, F. W., "Requirements for XML Document Database Systems". 10/11/2001. Atlanta, Georgia, USA
- [30]. Megginson, D., "SAX: A Simple API for XML". 2001.
- [31]. Hors, A.L., Hégaret, P.L., Wood, L., Nicol, G., Robie, J., Champion, M., Byrne, S., "Document Object Model (DOM) level 2 core specification". 2000. W3C Consortium.
- [32]. Bobkowaska, A., "Quantitative and Qualitative Methods in Process Improvement and Product Quality Assessment". 2004.
- [33]. Kemerer, C. F., "Reliability of Function Points Measurement". 1993. Communications of the ACM.
- [34]. Tigris.org: Open Source Software Engineering. *ArgoUML - A UML design tool with cognitive support*. <http://argouml.tigris.org/>.
- [35]. Eclipse Foundation. *Eclipse IDE*, <http://www.eclipse.org/>.
- [36]. Sourceforge.net. *Metrics Eclipse Plugin*, <http://metrics.sourceforge.net/>.
- [37]. Sourceforge.net. *eXist - Open Source Native XML Database*, <http://exist.sourceforge.net/>.
- [38]. The Cactus Test Framework, <http://jakarta.apache.org/>

**Fabio Perez Marzullo** is Marzullo Soluções em Gestão Empresarial Ltda. CEO, and became a graduate student at Federal University of Rio de Janeiro in 2003.

Fabio woks for the Brazilian Government and develops taxes and knowledge management software projects. He also manages a number of development projects, for some of the major Brazilian petroleum and communications companies, in the database laboratory at the Federal University of Rio de Janeiro, Rio de Janeiro, Brazil.

**Geraldo Bonorino Xexéo** is with Federal University of Rio de Janeiro.

Geraldo manages a number of development projects in the database laboratory at the Federal University of Rio de Janeiro, Rio de Janeiro, Brazil.