Providing Automated Support for the GQM Measurement Process

Automating support for the Goal Question Metrics process can cut data collection costs by exploiting data available in configuration management systems as well as data supplied by code analyzers. The author introduces a tool designed for this purpose and reviews its development and application in industry.

Luigi Lavazza, *CEFRIEL and Politecnico di Milano*

Measurement is a key factor for managing and improving software development. The measurement process aims to define and operate a measurement program (that is, a context-specific set of metrics) and to describe the required guidelines and procedures for data collection and analysis. Effective and efficient measurement requires methodological and technological support.

This article illustrates the automation of the most expensive phases of a Goal Question Metrics (GQM) measurement process. In particular, I describe the use of a specific tool for the definition, collection, analysis, and feedback of measures, and the techniques to interface such a tool with a configuration management system and metrics tools. (See the "Research and Practice" sidebar for more background information.) A measurement program at Pirelli Cavi, the Italian division of Pirelli Cables and Systems, has employed these techniques to automate the extraction of reliable and process-conformant data for the development of a Telecommunications Management Network (TMN) product. Although I focus on the GQM methodology, this work also provides general indications that apply in other contexts.

Why GQM?

The GQM[1] is a systematic technique for developing measurement programs for software processes and products. The GQM process is based on the idea that measurement should be goal-oriented—that is, data collection based on an explicitly documented rationale. You can apply the GQM process wherever you require a systematic approach to defining a measurement program.[2]

Figure 1 shows relative costs of the GQM's required main activities.[3] The absolute values are not valid in general because they depend on the nature and size of the goals and on the amount, nature, and quality of the required data. Nevertheless, they indicate these costs' relative relevance. Figure 1 reports both one-time costs (such as introducing the GQM method in an organization) and costs that occur on every execution of the GQM process. Producing GQM plans is considered a "per project" cost, although you can often reuse GQM plans, which reduces the process's cost. However, the degree of reusability (and savings) depends on the strategic goals.

Pirelli Cavi produces, among other products, TMN products compliant with CCITT (Consultative Committee for International

0740-7459/00/$10.00 © 2000 IEEE

Telegraph and Telephone) recommendations. Pirelli Cavi's management wanted a better understanding of their T31 TMN product's quality, regarding failures and faults, and its relation with the development process, regarding possible process improvement initiatives. (Here, I use IEEE standard terminology, thus the term *fault* indicates an error in the product, and *failure* indicates a perceivable problem caused by one or more faults).

In practice, the management's strategic goal was to determine the most critical fault-generation phases through an objective and quantitative characterization of the development process. Similarly, they wanted to identify the most critical fault-generation product parts. Therefore, Pirelli Cavi's management charged CEFRIEL (Center for Research and Education in Information Technology, www.cefriel.it), which already had experience in GQM application, with establishing a measurement process and the suitable support environment.

Following indications from Pirelli Cavi's management, I defined two GQM goals regarding software development in the R&D department:

- *Analyze* the development process of T31, *for the purpose of* better understanding the causes of faults, *from the viewpoint of* the software development team, *in* the Pirelli Cavi R&D department.
- *Analyze* the T31 product components, *for the purpose of* better understanding the origin of faults, *from the viewpoint of* the software development team, *in* the Pirelli Cavi R&D department.

The T31 product's development exploits PCMS (now Merant's PVCS Dimensions) configuration management software. PCMS, which uses an Oracle relational database as a repository, enforces a relatively rigid logical schema. PCMS provides customizable document templates, which can represent almost any kind of information. For the T31 product, the Change Requests and Test Defect Reports functions were derived from the templates to represent changes according to a given life cycle. These documents contain most of the data that the measurement plans need.

Automating the GQM Process

GQM plans can easily grow into quite complex and large graphs—a single goal can generate more than 100 metrics. Without specific automated support, keeping this amount of metadata under control is a challenging task.

The GQM tool

CEFRIEL developed the GQM tool, which runs in the Microsoft Windows environment, to overcome the problems connected with the management of GQM plans.

The tool supports editing of GQM goals through predefined forms. Explicit relations connect all of the plans' components (goals, abstraction sheets, questions, and metrics). The GQM tool can verify the structural consistency of plans—for example, it can check whether each question is connected with a goal and refined into metrics.

The GQM tool also allows reuse of the plans' components. The opportunity to reuse components of GQM plans occurred frequently in past applications. When defining a set of goals concerning the same development process or similar products, one component is commonly needed in different places (for instance, the same question asked in different goals). This is especially true in mature organizations, where developers share an explicit common model of the development process.

To make the interpretation and analysis of collected data feasible, it is necessary to provide the means to establish, maintain, and navigate links among data and the GQM plan. Using the GQM tool, it is possible to associate a GQM plan with an Access database and a GQM item with a query (or a table) defined in the database. The existence of these links makes the GQM items more understandable, modifiable, and reusable.

Analysis and interpretation of collected data form an important phase of the GQM process. This activity proceeds bottom up: collected data are analyzed and aggregated to derive answers to the questions and to evaluate the degree of accomplishment of the related goals. The GQM tool supports data analysis by running the queries associated with the GQM items and displaying the results. The GQM tool borrows the computational power from the database management

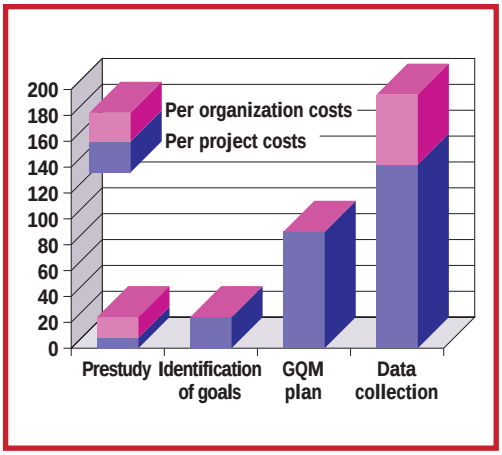

Figure 1. Cost of the Goal Question Metrics process phases.

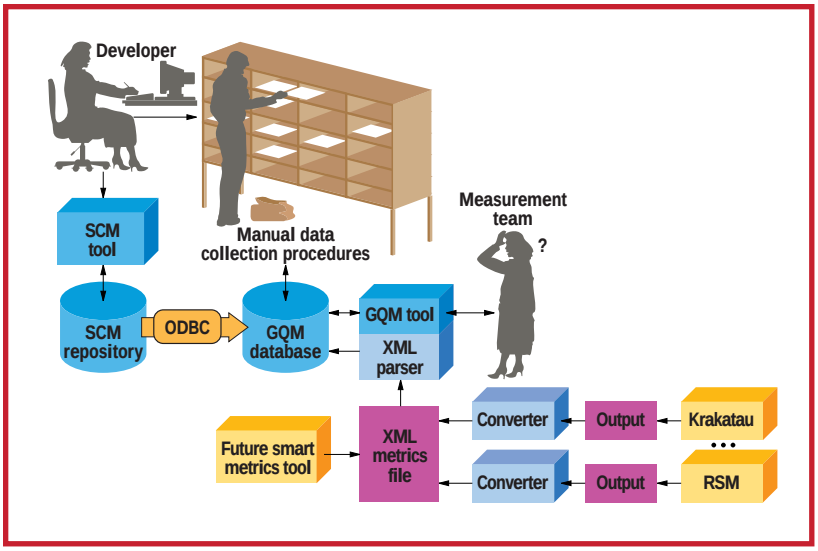

Figure 2. Goal Question Metrics integration with software configuration management and metrics tools.

system's query system. This is quite different from Metriflame, which provides data manipulation facilities in the tool (see the "Research and Practice" sidebar).

Applying the GQM tool

The GQM tool was used in the process of defining the GQM plans according to the goals previously described. As is often the case, in the work presented here an experienced measurement team applied the tool, rather than the process owner (that is, Pirelli Cavi's personnel).

Of course, the GQM plan had to refer to the development process of the Pirelli Cavi R&D Department and to the T31 product. Fortunately, both the process and product model were already represented in the SCM (software configuration management) tool used for the development. It was then quite easy to adopt these models as a conceptual reference to define the GQM plans.

For space reasons, I cannot report the whole GQM plan here. However, to give an idea of the plan, the following summarize the most relevant questions:

- What is the distribution of failures per priority?
- What is the distribution of faults per component?
- What is the distribution of faults per type and originating phase?
- What is the distribution of faults per severity?

The associated metrics included:

- Total number of failures;
- For each failure, priority (that is, how bad the effect was for the user), type, detection time, conclusion time, and so on;
- Total number of faults;
- For each failure, severity (that is, how difficult it was to correct), type, phase when originated, detection time, correction time, and so on.

Configuration management data collection

PCMS stores data in an Oracle database. A set of views (generated through a SQL script distributed together with PCMS) lets users access these data. The GQM tool uses an Access database as a repository. To let the GQM tool access PCMS data, it sufficed to make Oracle data visible as an Access database through ODBC (Open DataBase Connectivity). In Access, we created a new table, called T31_DATA, and then we linked it to the relevant Oracle view. Any access to T31_DATA was thus transformed into an access to Oracle data. Once the link was established, no further operation (such as data import and conversions) was needed: data created in PCMS was automatically available in the Access database, and hence to the GQM tool.

My coworkers and I automatically derived most of the required data from the PCMS repository. Among these, several data concerned failures (type, priority, and so on) and faults (severity, responsible component, and so on). However, some data required by the GQM plan could not be collected automatically, including the experience and domain knowledge of the personnel involved in fault detection and correction. Noticeably, data concerning the effort for correcting faults (both the estimation and the actual value) were already present in the documents stored in PCMS and were easily retrieved. In this case, the manual data collection activity already formed part of the normal development process, and the measure- ment process simply exploited it.

The data contained in T31_DATA were connected to the corresponding metrics definition in the GQM tool through suitable queries. All queries were simple and required little time to be written.

The data integration technique presented here applies to any SCM tool employing a relational DBMS repository supported by ODBC. In practice, this technique can apply to most commercial SCM tools.

Metrics data collection

Although the GQM tool can automatically access data in an SCM tool, such data doesn't always carry all the required information, such as the code's size and complexity. The SCM tools store source code, but you would need to compute its properties.

Several programs can measure static code properties. Nevertheless, you cannot instruct these tools to yield results conforming to a well-defined schema (in our case, a GQM plan). We further extended the GQM tool to take advantage of the data that software metrics tools produce. Particularly, we considered Krakatau and Resource Standard Metrics (RSM). These tools produce different outputs, such as number of SLOC, cyclomatic number, number of classes, methods per class, and so on. Both of them produce HTML output, but quite surprisingly none yields XML output. Because XML is rapidly becoming a standard for the representation of CASE tools data, it seemed quite natural to organize the GQM tool's integration with metrics tools in two steps:

- We defined a DTD (document type definition) for representing code measures. The GQM tool was equipped with a parser that reads XML files that use such a DTD and stores the corresponding data in the Access database.
- For each considered metrics tool, we developed a translator from the native format into XML.

We justify our choice of introducing an intermediate data representation in XML by considering that when metrics tools yield results in XML, the GQM tool will be ready to understand them. We might see several metrics tools producers adopt a unique DTD (probably similar to the one that the GQM tool is using) in the future; in that case, the GQM tool could read the output of several metrics tools with little or no translation.

Figure 2 illustrates the GQM tool's integration with SCM and metrics tools.

We found it relatively easy to solve the problem of translating and importing data. We faced a more subtle problem in deciding where to store the data from the metrics tools, because every piece of data could correspond to different metrics, according to the GQM plan. The following set of tables, which can contain any outcome of the metrics tools, gives a solution:

- Table "Component" stores the list of items (projects, files, classes, methods, and so on) being measured.
- Table "Composition" indicates the hierarchical decomposition of items (a method belonging to a class, defined in a file, or belonging to project).
- Table "Metric" stores the list of metrics (LOC, lines of comments, cyclomatic number, and so on). These are metrics as defined by the tools, not by the GQM metrics.
- Table "Measure" stores the values, which are related to a metric and to a component.

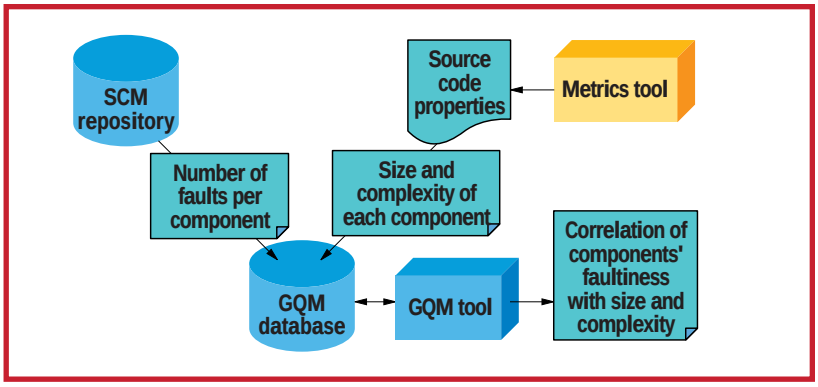

Figure 3. Use of data from different sources.

Exploiting multiple data sources

We can see that the combination of the data made available by the integration with the SCM tool and the data extracted from the metrics tools permits sophisticated analysis. For instance, we obtained the number of faults per component from the SCM tool and the complexity of every method from the metrics tools. We can then easily compute each component's average complexity and size based on the methods' complexity and size and the known composition relations. Figure 3 shows how we can analyze the correlation of the number of faults with both size and complexity.

We did not apply data extraction from metrics tools at Pirelli Cavi, because the tool's required extensions became available too late. Instead, we tested it in a research project at CEFRIEL. In this project, the effort dedicated to manual data extraction and conversion was about one personhour for every 300 Kbytes of code. The same work using the GQM tool's new facilities took just a few minutes for the whole project code. In large real-world projects, the saved effort could easily be a relevant fraction of the develop-

This sidebar provides more information on various tools and environments related to the Goal Question Metrics process.

Software Metrics Tools

Several commercial tools (see www.pitt.edu/~ddarcy/isprof/metrics.html) support measurement of source code static properties (such as the number of lines of code or the cyclomatic number), both for traditional and object-oriented programming languages. Such tools typically perform a fixed set of measurements and rely on the file system to store results. Often results are generated in relatively easy to read or browse formats (such as formatted text or HTML).

In practice, these tools can easily extract metrics from code, but they do not typically support any methodology nor any data interpretation. The user must know why the extracted measurements are needed and how to treat them.

NonGQM Measurement Environments

The Emerald (Enhanced Measurement for Early Risk Assessment of Latent Defects) environment was designed to improve software reliability in the telecommunications domain.[1] Emerald supports measurement of software code and analysis of collected data according to given quality models. Particularly, Emerald incorporates the Datrix code metrics tool for measuring static attributes of source code such as size, complexity, structuredness, readability, testability, and so on. Emerald's approach mainly differs from GQM's in two ways:

- It adopts an ad hoc methodology (particularly suited for the telecommunications industry).
- It pursues a fixed goal (software reliability), requiring a fixed set of metrics and sources (the code). GQM can be applied to any goal, involving any set of metrics, including Emerald's, or data sources.

Squid is a methodology and tool to plan and control software quality during development.[2] Although measures play an important role in Squid, the method focuses on defining the quality model for the considered software product. Squid requires a model of the product and the process and that these models be linked to the quality model. Measures apply to the attributes of the product and process that affect quality characteristics. Thus, Squid provides a clear framework to support software measurement. However, the approach draws on traditional measurement and analysis methods.

GlobalManager (www.globalmanager.org) collects and displays data generated by software development. Here, the emphasis is on the fast availability of data to support project management. To achieve this goal, GlobalManager imports data from several development tools. However, GlobalManager does not offer methodology guidance; it just provides data and facilities to display and analyze them, leaving the user with the task of data manipulation and interpretation.

Configuration Management Tools

Configuration management tools are valuable data sources. In fact, modern SCM tools not only store products of the development process, but they also tend to be aware of their state and state transitions, development activities (such as product building), and communication (for example, notification of relevant events). All these data are very important to quantitatively characterize the development process.

Nevertheless, SCM are not normally equipped with data collection facilities. The user must extract the needed information from the underlying repository—typically a commercial

ment time. You can find technical details not reported here for space reasons elsewhere.[4]

Results

Pirelli Cavi received valuable indications from the measurement process described in this article. However, I focus on the measurement tool and process, not on the defined metrics, the data, or the data analysis's potential consequences. I've outlined several interesting results concerning the process.

A more efficient process

The techniques described earlier permit automatic extraction of a relevant fraction of the required data. This implies that (after a simple set-up) the cost of data extraction was virtually null. Of course, this applies only to the metrics that you can extract from the SCM tool, the metrics tools, and other ready-to-use sources. You must obtain the remaining data by other means (for example, interviews), which are less amenable to automation. The savings derived from automatic data collection affect the measurement campaign's overall cost to an extent that depends on the nature of the GQM plans.

It is important to underline that automating data collection not only saves effort, but guarantees collection timeliness and accuracy.

A tool's availability also impacts the process. Early GQM plan formalization allows prompt goal verification and evaluation. Moreover, as knowledge about the observed process and product increases, you can easily modify the GQM plans, while keeping control of the plan's completeness and consistency.

Online feedback

Feedback sessions form a vital part of the GQM process. The GQM team meet with developers to agree on data interpretation

RDBMS (relational database management system). Reporting facilities often provide the user with limited support in this task.

GQM-Specific Tools

The first users of the GQM method soon discovered that developing GQM plans can be quite complex. Therefore, the first tools that provided specialized support to the GQM method (such as GQMaspect, GQM DIVA, and TEAM) focused on defining GQM plans. Later, some of these tools evolved to let users simultaneously view definitions of the metrics and their values, allowing a first rough interpretation of the data.

Most GQM-specific tools interface with commercial database management systems for data storage. First-generation GQM-specific tools did not pay special attention to data collection; they expected that someone would eventually feed the database with the proper data. Figure A depicts this situation.

More modern GQM-specific tools such as MetriFlame[3] can import data from different data sources through source-specific converters. A converter is a stand-alone program that connects to the desired data source (for instance, Lotus Notes), reads the data, and saves them into a MetriFlame-compatible format. Although this mechanism saves the time needed to manually convert and enter the data, the possibility of effectively decreasing the GQM process's cost depends on

- whether the data were entered manually in the original location for the GQM plan, or were already available for other reasons and
- how much elaboration is needed to fit retrieved data in the GQM plan.

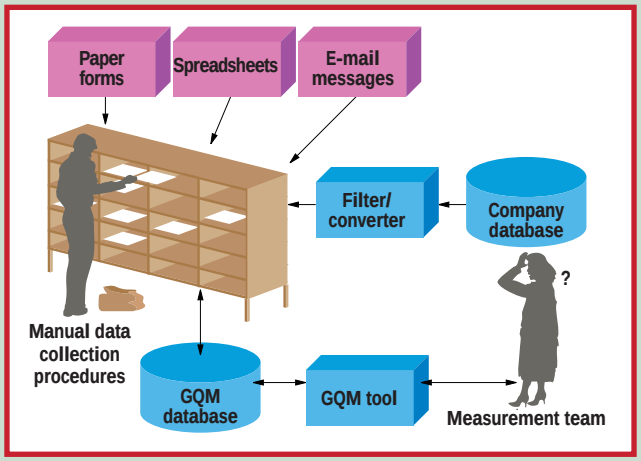

Figure A. Traditional data input.

What generally distinguishes the GQM tool is its current ability to manage both metrics definitions and the corresponding data. This function permits using a consolidated methodology for defining flexible measurement plans while data are made accessible from the development tools in a reliable, ordered, and economic way.

References

1. J.P. Hudepohl et al., "Emerald: Software Metrics and Models on the Desktop," *IEEE Software*, Vol. 13, No. 5, Sept. 1996, pp. 56–60.
2. J. Bøegh et al., "A Method for Software Quality Planning, Control, and Evaluation," *IEEE Software*, Vol. 16, No. 2, Mar./Apr. 1999, pp. 69–77.
3. P. Parviainen, J. Järvinen, and T. Sandelin, "Practical Experiences of Tool Support in a GQM-Based Measurement Programme," *INSPIRE II, Process Improvement, Training and Teaching for the Future*, C. Hawkings et al., eds., The British Computer Society, Wiltshire, UK, 1997.

and to verify consistency of the GQM plan with the actual development process. Meeting preparation and participation requires a substantial effort, which limits the frequency of feedback sessions (typically a half-day meeting every six to eight weeks). Thus, interesting results might be disclosed several weeks after they become available.

The mechanisms presented earlier permit faster and cheaper feedback sessions, because of the immediate availability of most data with negligible cost. Moreover, the risk of overlooking interesting data is almost eliminated by the possibility of specifying consistency queries: for example, data that do not conform to the baseline hypotheses are natural candidates for discussion. Quality targets in Squid (see the "Research and Practice" sidebar) offer a similar functionality.

In the best case, developers could use directly the GQM tool to browse as soon as metrics' values become available. However, traditional feedback sessions are still necessary whenever several people need to discuss the interpretation of collected data.

Increasing GQM process effectiveness

The GQM process has been criticized for being not repeatable and for being nonterminating.[5] Using supporting tools helps in this respect. For instance, with the use of a tool, you can quite naturally develop a library of goals, questions, and metrics that are consistent with the given environment and that you can reuse across the organization—thus helping the convergence toward common consolidated measurement programs.

Moreover, reusing existing collections of questions and metrics helps limit the measurement plan's scope. For instance, you can attach documentation to a question explain-

ing why the current set of associated metrics suffices or reminding which other metrics have failed to provide meaningful results.

The underlying database is a valuable reference point for termination. In fact, refinement of the GQM plan can stop when you can link every metric to a database query. Thus, you can assess the feasibility of metrics against the database, whose schema must in turn be consistent with the conceptual model of the software artifacts and with the software process being considered.

Increasing metrics acceptance

Several authors have documented the problems that arise in the practical application of metrics in industrial settings. For instance, the Software Measurement Guidebook developed at the Software Engineering Laboratory remarks that data collection should be a minor activity; that measurement data are fallible, inconsistent, and incomplete; and that personnel treat measurement as an annoyance, not a significant threat.[6] Shari Lawrence Pfleeger reports from her experience at the Contel Technology Center: "metrics are welcome when they are clearly needed and easy to collect and understand."[7]

Adding automated support to the GQM process addresses the observations reported earlier:

- Much data is collected automatically, with no required effort.
- Limited human intervention minimizes the errors in data collection due to annoyance, oversights, imprecision, and so on.
- Metrics to be collected are defined precisely—both at the conceptual level (using GQM notation) and at the data level (using SQL).
- The focus is on the measurement results (which are available cheaply and timely), not on the collection of data, which is largely automatic.

This technique helps make the GQM process cheaper and more repeatable, systematic, and widely applicable. It satisfies several of the conditions Ross Jeffery and Mike Berry[8] identified for the success of metrics programs.

In the work presented here, the definition of the GQM plan, the process modeling, and the product modeling proceed in parallel, using independent tools. The GQM team had to keep the GQM plan definition consistent with the models. A possible future improvement might be to integrate the GQM tool with modeling tools to make it easier to maintain consistency.

I did not give much attention to precisely assessing the automated GQM process's cost in this article. From a qualitative point of view, the tool's benefits in terms of savings, data reliability, and general acceptance of the measurement process are clear. Particularly, in this case, the measurement program would hardly have been possible at all without automated data collection because the management would not have accepted a regular GQM process's cost. Future controlled experiments will provide a precise quantification.

Another minor limitation of the presented approach is that the user must manually launch the measurement programs. Future GQM tool enhancements will let the user control the metrics tools through the GQM tool.

Acknowledgments

I thank the people at Pirelli Cavi for contributing essential information on the T31 product and its development process. Thanks to the many students who helped with the software's implementation required to carry out the described work. The GQM tool's first release resulted from several persons' effort, including Alfonso Fuggetta, Sandro Morasca, and people from Compaq Italy.

References

1. V. Basili, G. Caldiera, and D. Rombach, "Goal/Question/Metric Paradigm," *Encyclopedia of Software Engineering*, Vol. 1, J.C. Marciniak, ed., John Wiley & Sons, New York, 1994.
2. R. van Solingen and E. Berghout, *The Goal/Question/Metric Method*, McGraw-Hill, New York, 1999.
3. A. Fuggetta et al., "Applying G/Q/M in an Industrial Software Factory," *ACM Trans. Software Eng. and Methodology*, Vol. 7, No. 4, Oct. 1998, pp. 411–448.
4. L. Lavazza, *Providing Automated Support for the GQM Measurement Process*, Tech. Report RT98005, CEFRIEL, Milan, Italy, Sept. 1998; www.cefriel.it/Se/Resource/metrics.
5. D.N. Card, "What Makes for Effective Measurement?" *IEEE Software*, No. 5, Nov. 1993, pp. 94–95.
6. *Software Measurement Guidebook*, Tech. Report SEL-94-102, Software Engineering Laboratory, NASA, Goddard Space Flight Center, 1995.
7. S.L. Pfleeger, "Lessons Learned in Building a Corporate Metrics Program," *IEEE Software*, No. 3, May 1993, pp. 67–74.
8. R. Jeffery and M. Berry, "A Framework for Evaluation and Prediction of Metrics Program Success," *Proc. First Int'l Software Metrics Symp.*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1993.

About the Author

Luigi Lavazza is a senior researcher at CEFRIEL (Center for Research and Education in Information Technology) and an assistant professor at Politecnico di Milano, Department of Electronics and Informatics. His research interests include advanced and process-centered software engineering environments, databases for software engineering, object-oriented technology, software process modeling, measurement and improvement, and requirements engineering. He graduated in electronic engineering from the Politecnico di Milano. Contact him at Politecnico di Milano, Dip. di Elettronica e Informazione, Piazza Leonardo da Vinci, 32, 20133 Milano, Italy; lavazza@elet.polimi.it; www.cefriel.it/~lavazza.