Software Process Measurement in the Real World: Dealing with Operating Constraints

Luigi Lavazza[1,2] and Marco Mauri[2]

[1] Università dell'Insubria – Varese, Dipartimento di Informatica e Comunicazione, Via Mazzini, 5 – Varese, Italy
[2] CEFRIEL, Via Fucini, 2 - 20133 Milano, Italy
{lavazza, mmauri}@cefriel.it

Abstract. Process measurement occurs in an increasingly dynamic context, characterized by limited resources and by the need to deliver results at the pace of changing technologies, processes and products. Traditional measurement techniques (like the GQM) have been extensively and successfully employed in situations with little or no operating constraints. This paper reports about a measurement project in which –in order to limit the cost and duration of the activities– the team could not perform ad hoc measurements, but had to rely almost exclusively on the data that could be extracted automatically from development and measurement tools already in use. Exploiting the flexibility of the GQM technique, and with the support of a tool supporting the GQM, it was possible to define and execute the measurement plan, to analyze the collected data, and to formulate results in only three months, and spending a very small amount of resources.

Keywords: Process metrics, Product metrics Goal/Question/Metrics, Process quality assessment.

1 Introduction

The work described here was carried out in an organization of Banca Caboto in charge of the maintenance of a dozen banking applications, consisting mainly of Java code, SQL code, and HTML code. The size of the applications ranged from about 30 KLOCs (300 Function Points) to over 500KLOCs (over 9000 Function Points). The maintenance process employed 41 full-time people (13 employees and 28 people hired from external organizations) organized in three groups, each coordinated by a maintenance team leader.

The management of the organization needed to perform some basic evaluations of the process and products in order to support estimation activities and decision-making. For this purpose –having realized that objective quantitative data were needed– the management had started two measurement initiatives. The first one aimed at measuring the static properties of the managed software. For this purposed they adopted the CAST tool (http://www.castsoftware.com/). Measurement of the code was performed every three months on the whole set of applications. The collected data included for each application: LOCs, number of artifacts, backfired function points, number of files,

Q. Wang et al. (Eds.): SPW/ProSim 2006, LNCS 3966, pp. 80 – 87, 2006.
© Springer-Verlag Berlin Heidelberg 2006

number of classes, average Java coupling and complexity, number of SQL artifacts, average SQL coupling and complexity, number of web pages. In addition, the difference between subsequent versions was assessed by measuring the variation of the aforementioned qualities. On the basis of these measures CAST computed a set of high-level indicators (most of which predefined), such as the "artifact granularity", functional size index, artifact coupling, technical complexity and standard violations. A second initiative consisted in measuring the Change Requests (CRs) stored in the tool adopted to keep track of changes (ClearQuest). The organization managed the CRs according to a standard lifecycle; transitions between lifecycle states were recorded by means of ClearQuest. The established measurement procedures provided the number of CRs per application and per state.

Although these initiatives provided the management with some useful data, they were not able to satisfy more complex evaluation needs, which the management expressed as a set of questions: *Are we doing our job well? Is the quality of the managed applications good? How good are the people in charge of maintenance?*

These questions were originated by the need to control, verify, estimate and evaluate the process and products, and ultimately to support management decisions.

The authors were asked to set up a measurement process that could deliver the required evaluation. It was thus decided to employ the GQM method [3, 4], which was suitable for converting the strategic goal into a measurement plan, and which had been previously successfully used by the authors [5, 9]. Throughout this paper we assume that the reader is familiar with the GQM.

The organization posed a few constraints that forced the GQM team to deviate from the standard GQM process. The constraints were:

- The measurement team had to provide results in three months. These could be initial results; however they had to be reasonably meaningful and reliable.
- The budget for data collection was quite limited.
- The measurement process had to be as non-intrusive as possible: the maintenance process was not to be disturbed. Only one project manager could be involved in the "manual" collection of data, and only for a very small fraction of her time.

The latter concern was originated by the need to keep the productivity of the maintenance process as high as possible –therefore people should not be distracted from their work– and by the awareness that the introduction of measurement programmes often generates resistance [7]: the management wanted to avoid problems with the acceptance of metrics programmes by developers.

The paper describes how the measurement activities were carried out in conformance of the constraints. We report what data it was possible to collect, how the original goals were affected by the limitations in measurement, and how it was necessary to redesign the GQM plan in order to fulfill the constraints.

The paper is organized as follows: Section 2 reports about the definition of the GQM plan. Section 3 describes the measurement phase; limits to the fulfillment of the GQM plan due to unavailable data are also described. Section 4 describes the data that it was possible to measure and the results that could be derived from such data. Section 5 illustrates related work. Finally, Section 6 summarizes the lessons learned and draws some conclusions.

2 The Planning Phase

The planning phase was carried out without taking into consideration any constraint. Although it was clear from the beginning of the work that only some of the required metrics were going to be collected, it was decided to build a complete GQM plan, i.e., a GQM plan that could in principle satisfy as thoroughly as possible the strategic goals. The rationale for this decision was twofold:

- It was not known in advance which metrics it will have been possible to collect. Excluding some metrics from the plan implied the risk of excluding metrics that actually could be collected without violating the operating constraints.
- The GQM team expected that the unconstrained GQM plan could provide a framework for assessing the relevance and quality of the available metrics, and for evaluating their meaning and reliability.

The strategic goals given by the management were translated –in a rather straightforward way– into the following set of GQM goals:

Goal 1: Analyze the maintenance process for the purpose of evaluating the quality of the product, from the point of view of the management of the organization.
Goal 2: Analyze the maintenance process for the purpose of evaluating the duration and cost of maintenance activities, from the point of view of the management of the organization.
Goal 3: Analyze the resources employed in the maintenance process for the purpose of evaluating their adequacy, from the point of view of the management of the organization.

The definition of the GQM plan was carried out employing the GQM tool [5, 10]. The tool supports the execution of GQM processes, by addressing both the generation of the GQM plan (including the precise definition of the metrics) and the successive phases of the process, namely data collection and analysis. The tool also integrates the measures database.

Given the very short time frame available for carrying out the whole GQM process, the availability of the GQM tool was fundamental. By employing the tool, the GQM team was able to define the GQM plan affectively and efficiently. In fact, in this phase the tool is particularly helpful in maintaining the GQM documentation in order, in identifying inconsistencies, redundancies and feasibility problems with the plan, and in generating the documentation for the management.

The GQM goals reported above were refined into abstraction sheets, questions and metrics according to the consolidated GQM practice. The complete GQM plan included 37 questions and 58 metrics.

The main object of the measurement activities was the execution and management of a Change Request. Therefore, most metrics concerned the CR. Every CR was characterized in terms of time and effort spent, type (defect correction or enhancement), lifecycle (i.e., the sequence of its states), application involved, amount and quality of the resources employed to perform the change, characteristics of the change (criticality, urgency, size, etc.).

3 The Measurement Phase

In order to produce reasonably sound and interesting results, while satisfying the constraints, the following operating criteria were adopted:

- Tools that were employed in the maintenance process had to be exploited to automatically derive as many measures as possible. This approach was expected to provide reliable data at a very low cost, and to provide measures as soon as the associated phenomena were available in the environment [9].
- Measurement tools that were already in place should have been exploited as well.
- Subjective data that did not require a big effort for collection (e.g., data that could be collected una tantum) were going to be obtained via interviews. For this purpose, the management designated one of maintenance team leaders to cooperate with the GQM team.

The analysis of the maintenance environment confirmed that the application of the criteria described above could result in deriving measures from CAST and from ClearQuest, and in obtaining some subjective data via interviews.

In order to ease the analysis phase, it was necessary to store all the collected data in a unique repository. ClearQuest records were initially extracted from the ClearQuest repository (currently implemented on top of an Oracle database) and inserted in a specifically designed Access database. All the measures corresponding to the GQM metrics were obtained by means of a step-by-step approach, which consisted of ad-hoc queries and some post-processing. In some cases the GQM team had to directly manipulate the contents of the tables. In the worst case, a simple Java program was needed to compute the relevant information concerning the durations of changes. The extraction of data from CAST was more difficult, since its internal repository was not designed (nor documented) in a way that allowed the final user to extract data from it. As a result, only some of the required data were extracted from the repository, while other data were obtained via the Web interface. Some of the data could not be obtained at all. Finally, all the collected data were inserted in an Access database, designed to store both measurements of code and data concerning difference between subsequent code versions. Differently from the CR information extracted from ClearQuest, no further post-processing activities were required.

When all the possible ways of extracting data from tools had been thoroughly explored, it appeared that the available data had a few quite serious limitations:

- The data was not at the required granularity level. In fact most of the metrics of the GQM plan were intended to capture the characteristics of each CR. On the contrary, the application code was measured every three months: thus the available data concerned versions that were “separated” by tens or hundreds of changes.
- It was not possible to retrieve the correspondence between every CR and the code modified in the execution of the request, since the ClearQuest records did not indicate which source files had been affected by the CR.
- Some fields in ClearQuest records were not regularly or consistently compiled. In particular, the indications concerning the estimated and actual effort required to manage a CR were often lacking or imprecise.

– Some subjective metrics were not collected, because the person that had to support the GQM team was too busy in her regular work to be able to dedicate enough time to the measurement activities.

As a consequence of these limitations it was quite clear that the original GQM plan could not be executed without modifications.

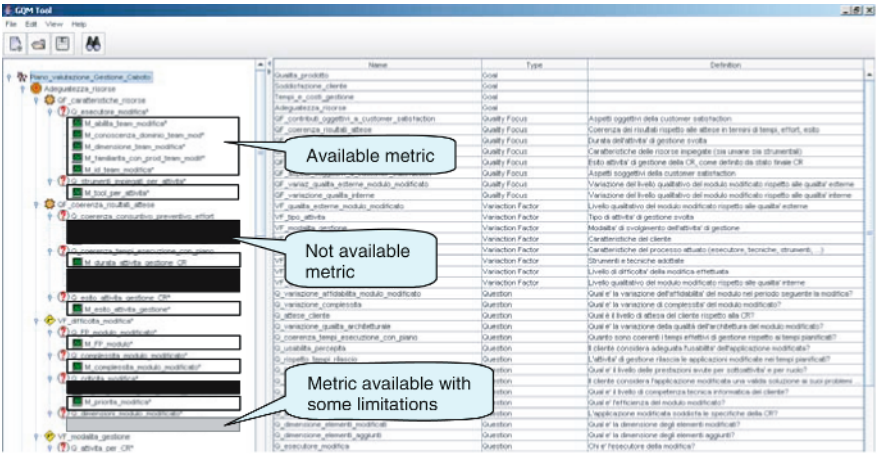

Fig. 1. Highlighting the available and not available metrics of the GQM plan

In order to understand the consequences of the unavailability of some metrics, and in particular in order to define a "simplified" GQM plan that could be successfully supported by the available metrics we proceeded as follows:

1. The metrics of the GQM plan were marked according to their availability. In Fig. 1 metrics are highlighted in different ways: boxed = available; blacked = not available; grayed = available with some limitations.
2. The structure of the GQM plan was exploited to understand the consequences of metrics unavailability: questions that had grayed or blacked metrics in their refinement could not be answered as planned. By considering the meaning of each gray and black element and its role in the GQM plan it was possible to assess to what extent the missing element could affect the goal.
3. On the basis of this assessment the whole GQM plan was revised in order to fit into the constraints. For instance the granularity of several questions changed: instead of referring to the management of the single CRs, they had to refer to the set of activities carried out in a three months period. As a consequence the involved goals did not change, but the associated results became less precise and accurate.
4. The GQM plan revision process was *dynamic*. In fact –also because of the short time available to complete the measurement process– it was not possible to fully understand what metrics were going to be collected before actually starting the collection phase. Therefore it was necessary to dynamically adjust the plan whenever a metric proved to be unavailable.

In summary, for goal 1, 14 questions out of 27 were modified and 2 were cancelled; for goal 2, 7 of 12 were modified; for goal 3, 4 of 13 were modified, and 2 cancelled.

4 Data Analysis and Results

The data collected from ClearQuest contained valuable information about the lifecycle of each CR. It was therefore possible to count the changes that were rejected, i.e., those that did not pass the acceptance test. It was found that the number of rejected changes was generally quite small, except for applications still under development.

Another type of analysis concerned the distribution in time of the CRs, according to their state. Fig. 2 shows the number of assigned, resolved and rejected changes per week. It indicates a generally good ability of the CR management process to satisfy the incoming requests, even in presence of peaks. However, it was not possible to estimate whether the volume of the work done to satisfy the CRs in a given time period was actually close to the amount required.

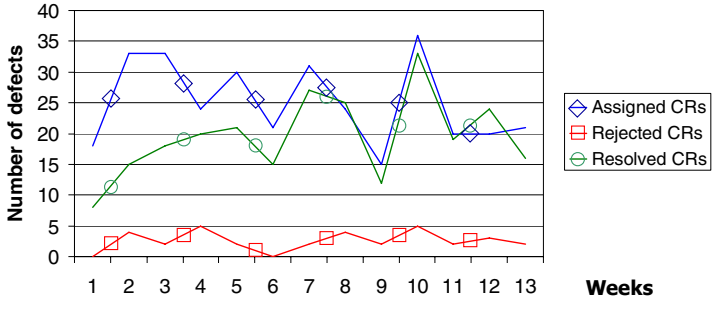

Fig. 2. Number of defect CRs per state in time

The GQM plan suggested also a type of analysis that was completely out of the scope of the previous measurement initiatives. In particular, the GQM plan indicated that the quality of the maintained product should be assessed in terms of defect density, which could be computed by combining the data derived from ClearQuest with the data derived from CAST. In particular, the dependency of the number of defects of an application from the characteristic of the application code was studied. It was thus found that no correlation could be established between the number of defects and the size (either measured in LOCs or in Function Points) of the applications. On the contrary, we found a good correlation between the number of defects and the number of Java classes contained in the code. In practice, data indicated that the object-oriented parts of applications were responsible for most defects.

Other results at the metric and question level are not reported here for space reasons. The results that could be obtained at the goal level are the following:

Goal 1. The maintenance process appears to be effective and the products of fairly good quality. Blocking defects and rejected changes are a small minority. Code changes do not affect quality.

Goal 2. Since it was not possible to collect measures on the "difficulty" of the CRs, nor on the effort required to perform changes, the part of the goal concerning costs could not be satisfied. The durations of maintenance activities appear reasonable and adequate with respect to priority.

Goal 3. The lack of data prevented the evaluation of the coherence between estimated and actual durations and costs. The resources appear generally adequate to satisfy the requests, preventing the creation of backlogs.

The results of the measurement process were presented (in much greater detail than given above) to the top management of Banca Caboto. They appreciated both the results and the method employed. They were also satisfied by the reusability of the measurement and analysis process and toolset in future measurement campaigns.

5 Related Work

Several experiences concerning "normal" usage of the GQM in industrial settings have been reported [5, 6, 13]. However, not much was reported about the usage of GQM in situations were data collection was severely constrained; in particular, we are not aware of any publication reporting the usage of the GQM as a tool easing the management of the operating constraints affecting the measurement process.

Actually, Mendonça and Basili [12] developed an approach combining the top-down GQM method with a bottom-up method based on a data mining. It is aimed at applying the principles of goal-oriented measurement in an environment that is already equipped with measurement practices. It aims at assessing if the user goals can be fulfilled by the data that is already being collected. Although this approach shares some objectives with ours, it is clearly more suitable for cases where large amounts of heterogeneous data are available. In our case, the identity and nature of the available data could be evaluated directly by the GQM team, who could assess whether user goals could be fulfilled by the available data and, when not, what modifications of the GQM plan were needed.

Concerning tool support, several articles address the problem of building frameworks specifically conceived to support measurement programmes [8, 2]. Unfortunately, it is often the case that a measurement programme has to be carried out in an environment that is not equipped with a suitable tool framework. Even worse, quite often the environment cannot be changed, or the allowed changes do not include the possibility of deploying new tools that could affect the development (or maintenance) process, e.g., changing the way developers (or maintainers) work.

Auer et al. evaluated tools that can be used in a measurement programme [1], but addressed rather low level issues, and considered only measurement tools, while other tools like ClearQuest can also play an important role as data providers.

6 Lessons Learned and Conclusions

A first observation is that tools (including development tools not specifically conceived for supporting measurement) can provide useful metrics. Data provided by tools –with the contribution of a small number of manually collected subjective data– can be sufficiently numerous and rich to support a whole measurement programme. Interestingly, tools provided the needed data in a quite non intrusive way.

In our case it was easier to extract data from a problem tracking tool than from a measurement tool: when selecting measurement tools, the possibility of exporting measures should be taken into due account.

The GQM tool was useful in organizing and documenting effectively the plan, and in supporting the identification of data unavailability and the evaluation of the consequences. For this purpose, the visibility "at a glance" of the plan, combined with the rigorous description of the GQM elements, greatly eased the task of revising the plan.

The GQM can provide a measurement framework that is useful even in presence of constraints that prevent several metrics from being collected. In the revision of the plan according to the data restrictions, the GQM was used –quite unusually– in a bottom-up fashion, as the decisions at the conceptual (goal/question) level were performed taking into account the situation at the operating (metrics/data) level.

In conclusion, the experience reported here can be seen as another confirmation of the value of the GQM, which performed well even in difficult and unprecedented operating conditions. Additional details on the work reported here can be found in [11].

References

1. Auer M., Graser B., Biffl S., A Survey on the Fitness of Commercial Software Metric Tools for Service in Heterogeneous Environments: Common Pitfalls, *9th International Software Metrics Symposium (METRICS'03)*, Sydney, Australia, September 2003
2. Aversano L., Bodhuin T., Canfora G. and Tortorella M., A Framework for Measuring Business Processes based on GQM, 37th Hawaii Int. Conference on System Sciences – 2004
3. V. Basili, GQM approach has evolved to include models, *IEEE Software*, vol.11, n.1, 1994.
4. Basili V., and Rombach H.D., The TAME project: towards improvement-oriented software environments, *IEEE Transactions on Software Engineering*, June, 1988.
5. Fuggetta A., Lavazza L., Morasca S., Cinti S., Oldano G., Orazi E., Applying G/Q/M in an Industrial Software Factory, *ACM ToSEM*, vol. 7, n. 4, October 1998.
6. Gresse C., Rombach D., and Ruhe G., Tutorial: A practical approach for building GQM-based measurement programs - Lessons learned from three industrial case studies, in Proceedings of *10th Brasilian Symposium on Software Engineering*, Sao Carlos (Brasil), 1996
7. Hall, T. and Fenton N., Implementing software metrics — the critical success factors, *Software Quality Journal*, Kluwer Academic Publishers B.V., vol.3, n. 4, December 1994.
8. Kempkens R., Rösch P., Scott L., and Zettel J., Instrumenting Measurement Programs with Tools, in Proc. *PROFES 2000*, Oulu, Finland, June 2000, F. Bomarius and M. Oivo Eds. LNCS Vol. 1840
9. Lavazza, L., Providing automated support for the GQM measurement process, *IEEE Software*, vol. 17, n. 3, May-June 2000.
10. Lavazza, L. and Barresi, G., Automated Support for Process-aware Definition and Execution of Measurement Plans, *ICSE 2005*, St. Louis, May 2005.
11. Lavazza, L. and Mauri, M., Measurement tool support in the real world: a GQM experience, CEFRIEL Technical Report RT06001, March 2006.
12. Mendonça M.G. and. Basili V.R,, Validation of an Approach for Improving Existing Measurement Frameworks, *IEEE TSE*, Vol. 26, No. 6, June 2000.
13. van Solingen R., van Latum F., Oivo M., and Berghout E., Application of software measurement at Schlumberger RPS, in Proceedings of *Sixth European Software Cost Modeling Conference*, Paris, 1995.