Experiences introducing a measurement program

S. De Panfili, B. Kitchenham, N. Morfuni

1Engineering Ingegneria Informatica S.p.A., Via dei Mille, 56, 00185 Roma, Italy
2Department of Computer Science, Keele University, Keele, Staffs ST5 5BG, UK

Abstract

Measurement is an integral part of total quality management and process improvement strategies. This paper describes our experiences using the Goal-Question-Metric (GQM) paradigm to help design a company-wide measurement program for Engineering Ingegneria S.p.A., an Italian software house. The introduction of the measurement program was supported by the Commission of the European Communities within the European Software and Systems Initiative (ESSI) as a Process Improvement Experiment (PIE). We found it necessary to supplement GQM into two ways. Firstly, we defined our measures rigorously in terms of entities, attributes, units and counting rules. Secondly, the original GQM plan was subject to an independent review. The most critical problem identified by the review was that the GQM plan identified too many productivity factors for any statistical analysis to handle concurrently. In order to address this issue, we developed an analysis technique based on a step-wise analysis of residuals. This has allowed us to identify the main factors affecting productivity and effort.

Keywords: Goal-Question-Metric (GQM); European Software and Systems Initiative (ESSI); Process Improvement Experiment (PIE)

1. The ENG-MEAS project

This paper reports experiences and results from the ENG-MEAS project (Introduction of Effective Metrics for Software Products and Processes in a Custom Software Development Environment). The project started in January 1996 and ended in June 1997. ENG-MEAS is funded by the European Commission ESSI programme which is an initiative for supporting technology transfer to industries.

1.1. Objectives

As a major Italian software house, Engineering undertakes many different types of software project including turn-key projects, time and materials projects, consultancy, training, software tool production. However, in the past few years large turn-key projects have become the most important line of business. This presents a problem since turn-key projects based on fixed contracts pose substantial commercial risks. Thus, the major business objectives of the company are to increase the profitability of, and to reduce the risks associated with, turn-key projects. In an effort to reduce these risks, Engineering needs to improve its ability:

1) to provide managers with effective support in prediction and control of software projects; 2) to ensure independent quality assessment and assurance for both the developed software and the software development process; 3) to have at its disposal indicators of the effectiveness/efficiently of the company’s quality system.

The company’s quality system has been certified compliant to ISO 9001 since January 1994. We believe that the establishment of a metrics database is crucial to its further evolution. The objective of the ESSI experiment is to establish a company-wide database of the most appropriate measures and also to identify the most effective and efficient procedures for collecting and analysing our data.

1.2. Starting scenario

1.2.1. Organisational environment

Engineering is a geographically distributed organisation, with local (currently, 11) units taking both commercial and technical responsibilities. Operational independence is counterbalanced by central support and control. Each local production unit reports to the central Production Department and is supported by the Central Technical Department, which is structured into four departments: (1) the Methodology Department, in charge of the maintenance and evolution of the in-house developed METHIS® software development methodology and of the provision of the associated
training to the company staff; (2) the Research and Development Department, in charge of assuring technological excellence and promoting innovation; (3) the Consultancy Department, in charge of furnishing specialised and highly skilled knowledge; (4) the Quality Assurance Department, in charge of managing the company Quality System.

1.2.2. Technical environment

The in-house METHISO® software development methodology and associated set of methods is usually adopted in the company's projects, unless different choices are imposed (e.g., by compliance to customer's life cycle and methods, by contractual obligations, etc.).

METHISO® identifies ten Macrophases in the software life cycle, grouped in three categories as shown in Table 1.

Specific control activities are performed during all these macrophases: (1) Project Management and Control; (2) Estimation and Planning; (3) Quality Assurance; (4) Configuration Management.

All technical staff are trained in METHISO®. In addition, the methodology Department is engaged in continuous revision and enhancement of the methodology in order to deal with new methods and techniques. The company has experienced a significant shift from traditional projects (development of mainframe-based, transaction-oriented applications, with languages such as COBOL) to projects for user-oriented applications, distributed in a network architecture and developed in languages such as C/C++, VisualBasic and/or by using CASE or RAD tools.

1.2.3. Cultural influences

As a result of Engineering’s de-centralised organisation, we expected some areas of resistance to the proposed measurement program:

- The relationship between central functions (mainly, methodology and quality assurance departments) and production teams sometimes results in friction and is heavily influenced by market conditions (e.g. the company’s quality system is sometimes viewed as a competitive advantage and sometimes as unnecessary extra work).
- The commercial and production functions have a joint responsibility during contract acquisition (e.g., estimation and costing) and in project management (e.g. customer management). This could make it difficult to analyse some relevant variation factors, such as schedule/resource constraints, customer characteristics, etc.
- Technical people have to be convinced that a measurement program is not intended to measure individual performance.

1.3. Workplan

ENG-MEAS is arranged according to the following plan, organised into four phases: (1) initial process assessment; (2) first definition of the innovation action; (3) data collection and analysis on baseline projects; (4) calibration of the innovation, based on the feedback from the data collection and analysis.

This structure of the project is intended to minimize the risk of a mismatch between the introduction of the measurement program and the real needs/capabilities of the company.

1.4. Expected outcomes

The main result of the project is intended to be the establishment of a company baseline to support a move towards increased quantification of our software development organisation.

In particular, we anticipate two direct outcomes:

- a more precise estimation and planning process, based on the availability of productivity data correlated to the more significant variation factors;
- clear evidence of problem sources and improvement needs, based on the availability of defect data correlated to the more significant variation factors.

Last but not least, a very important organisational outcome is the reduction in the degree of subjectivity involved in project evaluation as a result of using a set of objective and mutually agreed measures. This is expected to produce a more co-operative and less confrontational relationship between quality assurance and production staff.
2. Initial software process assessment

The initial assessment was conducted to obtain a baseline with which to compare final results and to focus objectives better during the subsequent phases.

The main results of this assessment can be summarised as follows:

2.1. Explosion of the technological environments used on the company's projects

In 1995 and 1996, we managed software projects involving more than 25 different technological environments (Operating Systems, DBMSs, TP Monitors, Languages, Development Environments) for software products ranging from mainframe applications to distributed client–server or intranet applications. This has two important consequences:

- it is difficult for project managers to use past experience in initial estimations;
- it is necessary to understand and control the spread of different technologies across Engineering's widely geographically distributed organisation (for example, we need to evaluate training costs, understand the overheads on projects resulting from the learning curve, exploit the repeat of usage of new technologies, and provide central support for a variety of new skills).

2.2. No systematic collection of data on the size of the developed software systems

Available data referred to lines of source code or to number of programs. Even if such measures were accepted by managers, they are not effective in an unstable environment (see previous point). Furthermore, more modern and less technology-dependent techniques, such as Function Points, were not well known to the technical staff and were not extensively used. A central staff function was established to act as initial skill centre on this topic and significant effort was devoted to recovering the Function Point measures for a large set of past projects. Steadily, function point knowledge is being transferred to project teams and this is allowing us to decentralise some measurement activities.

2.3. Criticalness of the correctness of productivity evaluation

Another critical issue that emerged was that attention was paid only to productivity of the projects until delivery. Turnkey projects, after delivery, have a guarantee period (typically 6–12 months) during which the company has to perform corrective maintenance on the delivered product. Obviously, if significant activities are performed during the guarantee period, overall productivity (and profitability) will be adversely affected. Unfortunately, the existing effort accounting procedures did not allow a precise measure of the development effort expended during the guarantee period.

2.4. No meaningful indications for defect rates

There were no standard and suitable measurements available to assess the quality of the development process and the effectiveness of each life-cycle phase, such as: number of defects, nature of defects, originating phase, finding phase, costs to fix, etc.

3. Definition of the measurement program

We believe the most successful way to determine what we would measure is to tie the measurement program to our organisational goals and objectives. The PIE selected the 'Goal-Question-Metric' method ([1–4]) as a means of defining and documenting the links between the measurements to the goals. The GQM paradigm has been developed as a systematic technique for developing a measurement program for software processes and products. GQM is based on the idea that measurement should be goal-oriented, i.e., all data collection in a measurement program should be based on a rationale which is explicitly documented.

The most important product of the GQM paradigm is the GQM plan, produced to relate the organisation's goals to the metrics to be used to evaluate their degree of accomplishment. A GQM plan is defined in terms of:

- **Goals:** a goal is defined in terms of: an **Object** (the part of reality that is being observed and studied—product, process, phase, etc.); a **Purpose** (the motivations for studying the object—to understand, to engineer, to control, etc.); a **Quality Focus** (the object characteristics and facets that are considered in the study—cost, productivity, reusability, etc.); a **View Point** (the person or group of people interested in studying the object—users, managers, developers, etc.); and an **Environment** (the application context where the study is carried out).

- **Abstraction sheets:** each goal is associated with an Abstraction Sheet which is composed of four parts: **Quality Focus** (additional details on the object characteristics); **Variation Factors** (process and product characteristics that may affect the quality focus); **Base level hypotheses** (current status of the object of study with respect to the quality focus); and **Impact on base level hypotheses** (how the variation factors are expected to affect the current state of the object of study).

- **Questions:** from the abstraction sheet a set of questions is derived in order to define and to characterise the goal in an operational way.

- **Metrics:** from each question a set of metrics is derived. The metrics are used to collect data in order to answer the questions.

Typically the same question may arise from the
refinement of different goals; in turn, the same metric may be used to answer several questions. Thus, after the initial effort for the definition of a GQM plan, the definition of new plans can be expected to reuse much of the initial work, saving time and effort.

In practice we found it necessary to extend the basic GQM process in two ways: we needed to extend some of the basic GQM activities and we needed to make explicit the context within which the GQM plan was being developed. These issues are discussed below.

3.1. Additional GQM activities

We added two extra activities to our GQM process: (1) systematic measurement definition; (2) independent review of the preliminary GM plan.

3.1.1. Systematic measurement definition

We used the framework proposed by Kitchenham et al. [5] to define each ‘metric’ more rigorously in terms of:

- **Entities**: the entity from which the measure was to be extracted (e.g. the development project).
- **Attributes**: the property of the entity to be measured (e.g. staff effort).
- **Units**: the means by which the attribute was to be quantified (e.g. staff days) and the scale type of the unit (e.g. ratio).
- **Counting rules**: the procedure to be adopted when the measure was extracted (e.g. the time at which the measure is made, who is responsible for extracting the measure, what tools, if any, are used to assist data collection).

When defining our measures, we were also aware of the need to identify which aspects of the measures were to be recorded. By aspects we mean actual measures, estimates and/or target values.

3.1.2. Independent review of the GQM plan

Producing a GQM plan is analogous to defining the requirements of a software product, and as such it should be subject to a review process. We used a process of independent review. The GQM plan was produced by Engineering staff and reviewed by one person with software metrics and data analysis experience who took no part in the development of the plan. The review was based on the following review criteria:

1. Internal consistency of the GQM analysis: this involves considering whether there are any aspects that have been overlooked and whether or not any questions and metrics are unnecessary in terms of the goals.
2. Data collection feasibility: this involves considering whether or not the identified measures will provide reliable data (i.e. whether the measures are defined well enough to be repeatable and comparable).

3. Analysis feasibility: this involves considering whether or not the identified measures can be analysed in a way that allows the basic goals to be achieved.

The review took 3 days to perform, including preparation of a review report. The results of the review was used to produce a revised version of the GQM plan.

3.2. GQM context

Our aim was to establish a valid common measurement framework that would remain valid for all of Engineering’s projects. Thus, the GQM plan was produced within the framework of the following guidelines and constraints:

- The purpose was to characterise the development process phases with respect to two main aspects: productivity and defectiveness.
- The analysis was focused on the company’s product line as represented by turn-key projects. Other product lines (time and material contracts; consultancy, etc.) were not specifically dealt with.
- The analysis was intended to focus on the technical aspects of development projects only, without exploring the economic dimensions of a project (costs versus effort; subcontracting, etc.) or the optimisation needs at the whole company level (allocation of staff on projects, staff under utilisation, etc.).
- With reference to METHIS©, the focus is on the macro-phases specifically devoted to software development (Conceptual Design, Technical Design, Realisation, System Test and Delivery, and Guarantee).
- The focus is on variation factors that might have a significant effect, typically those that are not restricted to a small part of the lifecycle (however, an important exception is represented by a factor such as a programming language which is likely to have a major impact on productivity).

4. GQM results

4.1. GQM analysis results

The basic GQM plane defined two goals:

<table>
<thead>
<tr>
<th>Goal 1</th>
<th>Goal 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object:</td>
<td>Development process phases;</td>
</tr>
<tr>
<td>Purpose:</td>
<td>characterise;</td>
</tr>
<tr>
<td>Quality Focus:</td>
<td>productivity at the delivery and at the end of guarantee;</td>
</tr>
<tr>
<td>Viewpoint:</td>
<td>management; project manager;</td>
</tr>
<tr>
<td>Environment:</td>
<td>company’s turn-key projects.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Goal 2</th>
<th>Goal 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object:</td>
<td>Development process phases;</td>
</tr>
<tr>
<td>Purpose:</td>
<td>characterise;</td>
</tr>
<tr>
<td>Quality Focus:</td>
<td>process defectiveness;</td>
</tr>
<tr>
<td>Viewpoint:</td>
<td>management; project manager;</td>
</tr>
<tr>
<td>Environment:</td>
<td>company’s turn-key projects.</td>
</tr>
</tbody>
</table>
Table 2
GQM quality focus and variation factors

<table>
<thead>
<tr>
<th>Quality focus</th>
<th>Defectiveness</th>
<th>Process factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Productivity</td>
<td>Defects created per macrophase</td>
<td>Development environment</td>
</tr>
<tr>
<td>Functionality</td>
<td>Defect criticalness</td>
<td>Customer characteristics (availability, acceptability of intermediate results)</td>
</tr>
<tr>
<td>of delivered systems</td>
<td>Defect found per macrophase</td>
<td>Development environment</td>
</tr>
<tr>
<td>Sources lines of code</td>
<td>Defect characteristic: Nature (req, design, cod, doc, ...)</td>
<td>Distribution of the effort on development macrophases</td>
</tr>
<tr>
<td>of delivered systems</td>
<td>Defect characteristic: Introduction phase</td>
<td>Software development lifecycle</td>
</tr>
<tr>
<td>Function point counts</td>
<td>Defect characteristic: Detection phase</td>
<td>Constraints on plans (time, resources)</td>
</tr>
<tr>
<td>of delivered systems</td>
<td>Defect characteristic: Detection technique</td>
<td>Development team</td>
</tr>
<tr>
<td>Programs of</td>
<td>Phase containment effectiveness</td>
<td>Test coverage</td>
</tr>
<tr>
<td>delivered systems</td>
<td>Ability to find and repair defects in macrophase that introduces them</td>
<td></td>
</tr>
<tr>
<td>Re-work</td>
<td>Rework cycles per macrophase</td>
<td></td>
</tr>
</tbody>
</table>

The analysis of the goals produced: 14 Quality Focuses (see Table 2); 12 Variation Factors (see Table 2) most of which apply to both productivity and defectiveness; 41 Questions; 68 Metrics.

The GQM plan was developed by a two-person team that had been trained by academics with extensive experience of the method. The team coordinated the planning activity with other Engineering staff including the Production Department manager, the Methodology Department manager, and the Quality Assurance Department manager. Managers of local production units were also been interviewed in order to gain a more detailed understanding of their specific requirements. The team took 2 months to produce the version of the GQM plan that was the object of independent review. The subsequent revision was performed by the central team and then discussed with the full group of participants.

The goal of the plan was intentionally 'ambitious': the experimentation phase, together with the planned parallel feedback activities, was intended to provide any necessary calibration and simplification.

The definition of the GQM plan was supported by a shareware tool\(^1\) - the GQM tool developed at CEFRIEL, Italy. Fig. 1 gives a fragment of the GQM View Hierarchy. The tool does not provide support for applying the GQM method. However, it allows the user to assess the completeness and consistency of the plan and, thus, was particularly useful for plan maintenance.

4.2. Review results

The independent review of the GQM plan identified a total of 43 queries against the plan [6]. These were classified by type: IC (Internal consistency); DCF (Data collection feasibility); and AF (Analysis feasibility) and severity:

- Critical (C): an issue that renders (or will render) the GQM analysis or any data collection scheme based on it unusable.
- Serious (S): an issue that affects a number of related metrics/questions.
- Minor (M): an isolated issue affecting a single question/metric.
- Negligible (N): an issue that if not corrected would have no impact on the effectiveness of the GQM analysis and/or any data collection system based on it.

A summary of the review statistics are shown in Table 2. In general, queries related to specific questions and metrics that were common to the productivity and defectiveness analyses were only reported once. The exception is the critical analysis feasibility issue. For both productivity and defectiveness the critical issue concerned the number of different variation factors. With 12 variation factors most of which were modelled as nominal or original scale with two to five scale points, it is extremely difficult to collect sufficient data to provide a rigorous statistical analysis. This observation led to the specification of a specialised data analysis procedure based on stepwise analysis of residuals [7].

To illustrate the type of issues that were raised, a few of the issues were as follows:

\(^1\)The tool is available via anonymous ftp from ftp://ercole.cefriel.it/pub/ Settore2/gqm.tool.
One metric was based on a count of functions. The review issue identified the difficulty of defining a function and ensuring that functions were counted consistently over different projects. This was classified as a Serious issue related to Data Collection Feasibility.

The method by which early phase defects were to be collected was not defined. This was classified as a Serious Data Collection issue.

The quality focus on productivity was not matched by any specific questions related to productivity values (only effort-related questions were asked). This was classified as a Minor Internal Consistency issue.

A binary yes/no metric was called ordinal instead of nominal. This was classified as a Negligible Internal consistency issue.

As already mentioned above, the review process results (see Table 3) were used to update the GQM plan. Examples of revisions include the deletion of 'Functionality' as Quality focus, better specification of counting rules, etc.

5. Data collection and analysis

At the point in time of writing this paper the Data collection activity is still in progress. The data analysis was performed on data set containing 29 projects.

Development effort and unadjusted function point data have been easy to collect because in the case of effort, procedures for collecting it were already in place, and in the case of unadjusted function points the PIE provided trained staff to extract the measures. Obtaining data items that were not collected previously, i.e. defect data and post-delivery effort has been more difficult. Indeed, to overcome problems, the project team established a specific data collection procedure that, in some cases, has also been adopted by our clients. Nevertheless, at present few projects have collected these data items. This fact emphasises the need for measurement programs to make use of existing data whenever possible, as well as the need for data collection procedures that can easily be updated as soon as the
analysis process reveals that there is insufficient information
to draw any conclusion.

5.1. Data analysis method

Data analysis concentrated on the definition of company productivity and defectiveness statistical baselines. In this context, a statistical baseline comprises the average values and variance of productivity or defect rates for projects developed by the company, allowing for the effect of significant variation factors. The baselines are intended to support:

- our ability to describe the company’s development capabilities to support project estimation and planning;
- the provision of a starting point against which the effect of process changes can be evaluated. This allows us to address many of the questions raised by the GQM plan.

In order to define the baseline, we have adopted a procedure for analysing unbalanced datasets that include many nominal and ordinal scale factors [7]. The main advantages of the procedure is that it handles measures of all scale types (nominal, ordinal, interval ratio and absolute); identifies factors with confounded effects; and ensures that the analysis procedure does not over-analyse the data, i.e., it uses degrees of freedom correctly. For implementing the analysis procedure (which is not capable of full automation), we are using a general purpose statistical package—WINKS by TEXASOFT.

In order to establish our statistical baseline, we are collecting data on 29 completed turn-key projects which use different target software and hardware technologies. This set includes projects aimed at delivering software applications, as well as products, and covers different application domains. At this stage the only constraint has been to exclude projects which are only concerned with maintenance of existing software applications.

In addition to establishing a baseline, we are also interested in identifying those projects which deviate significantly from other projects either by being extremely good or extremely bad. For this purpose we use a Bivariate Anomaly Detection analysis performed by placing a grid over a scatterplot derived from project productivity values and identifying data points that have no neighbours in the same and surrounding grids. From the point of view of anomaly interpretation, anomalies can either identify good practices to follow in future projects, or bad examples from which to identify problems and risks.

5.2. Analysis results

Because data collection is still on-going, we only have some preliminary results concerning development productivity. Analysis of project productivity has identified that the main statistically significant factor affecting productivity is development environment generation (third, fourth, and KAD). The relationship between productivity and development environment generation is highly significant ($p < 0.001$) and development environment generation accounts for 47% of the variation.

In accordance with our adopted analysis method, we looked for the second most important variable. At this step we wound three significant variables: final profit, $p = 0.016$; coding as a percentage of development effort, $p = 0.036$; square root of total effort, $p = 0.04$.

Since profit is not really an independent variable and the effect of coding percentage was counter-intuitive (and therefore possibly spurious), we decided to investigate the effect of including total effort as the next variable in our model. This line of analysis lead to the identification of size as the third important variable (measured as Unadjusted Function Points), and the fourth and last statistically significant variable was the unit site. The model based on these four factors accounts for 83% of the variation.

This analysis must be treated with some caution not only because of the limited amount of data we have collected as yet, but also because size and effort are functionally related to productivity, so there is a danger that we may have detected spurious relationships.

To provide some check on our productivity model, we performed a similar analysis using effort as the dependent variable. This analysis revealed that only size (55% of the variation) and development environment generation (10% of the variation) have statistically significant effects on effort. Considering both analysis together, these results suggest that only development environment generation has a real effect on productivity.

During our data analysis work, we identified two problems:

(1) At the start of the analysis activity, it was intended to restrict analysis to projects classified as complete. Initial analysis of productivity values identified one project as having an unusually high productivity value. Further investigation of the data identified that it was a requirements specification project only—no implementation was undertaken. This has alerted us to the need to ensure that ‘completed projects’ are genuinely comparable in terms of their work content and outputs.

(2) Preliminary analysis of partial datasets can be misleading. Our initial analysis of productivity based on a subset of our data suggested a possible economy of scale (i.e. larger projects were more productive). However, with a larger dataset this effect seems to be less significant, and productivity appears to be unaffected by product size.

The bivariate anomaly detection has been supported by the use of a tool produced by the ESPRIT SQUID project [8]. It has proved useful in understanding more about the nature of productivity improvements. Fig. 2 shows some graphical results of the Anomaly Detection as performed by SQUID. It identifies projects PUU02, MUB03, PCT15 and HPR01 as anomalous. In this case possible reasons are:
PCT15, HIPR01: both projects used Magic© by MSE. The high productivity suggests that it is a powerful RAD environment. It is also interesting that some projects produced using Magic© did not have particularly high productivity. One project (PCT08, the project closest on the left to PCT15 on the same row—but not identified as anomalous) was produced by the staff who afterwards developed PCT15. Like PCT15 it used Magic© and it was of almost the same size, however it was not significantly more productive than many conventionally developed products. We interpret this to be an example of a learning curve effect. It implies the need for a training period, although relatively short, when adopting the Magic© environment.

- PUU02 had a relatively low productivity for its size. Investigation of the project indicated that the low productivity was due to very stringent documentation and user validation requirements, which meant the project went through a very formal and documented development process. It was performed with a strict adherence to METHIS© in a classical structured analysis environment adopting COBOL and DB2©.
- MUB03 had relatively high productivity for its size, in spite of being based on COBOL and ORACLE©. The high productivity appears to be due to extensive reuse of already existing software.

Our experiences so far suggest that a formal statistical analysis is useful for identifying widespread trends while anomaly detection is useful for identifying rare, non-systematic effects that would not be detected by formal statistical analysis. However, if anomaly detection is to be used successfully, it is important to be able to obtain additional information about the likely causes of anomalies from the project staff.

6. Initial assessment of the PIE

A complete assessment of the impact of the measurement program on the business will not be possible until the end of the measurement program when a cost/benefit analysis will compare the costs associated with the measurement program, such as:

- effort for quality assurance staff
- overhead for project managers and teams and the obtained results, such as
6.1. Organisation

The most obvious organisational impact is a reinforcement of the methodology and quality assurance functions, as a result of their role in collecting, validating and analysing measures.

6.2. Culture

Until now people have reacted positively to the PIE, viewing it as a significant step towards a quantitative decision-making process. Operating in a rapidly evolving business and technology environment, people are increasingly conscious of the risks associated with a lack of effective software data. Indeed, consensus has been reached on the aims of the measurement program and on the achievement of a set of measures consistent with such aims.

Future work has to confirm the present positive reaction and overcome any remaining resistance (e.g., by confirming that the measurement program is not intended to measure individual performance).

6.3. Skills

Engineering has benefited from the acquisition of specific measurement-related skills. Now we have available a set of people ('measurers') able to define and implement a measurement program according to a well-established methodology. An example is Function Point counting. This technique was not well-known to the technical staff. Thus, in order to assure reliable and comparable measures from the start of the experiment, FP collection was centralised. However, as training on the subject has progressed, the measurement activity is being decentralised to individual project teams.

6.4. Strengths and weaknesses of the PIE

At present, the major strengths of the adopted approach are:

- adoption of a precise methodology for the definition of the objectives of the experiment, i.e. for the definition of the measurement program;
- emphasis on objective measures in order to increase data reliability, and comparability.

On the other hand, some weaknesses are also evident:

- at the present state of the art in the metrics field, it is difficult to obtain a significant coverage of all factors of interest by means of objective measures;
- the low involvement of the commercial function is associated with a focus of the experiment on the technical aspects of the software development process. This means we have only achieved a partial coverage of the company's quality system measurement needs.

6.5. Summary of lessons learnt

With respect to the particular technologies we have adopted, our experiences to date can be summarised as follows:

- The top-down, systematic approach enforced by GQM is effective in helping to achieve an internally consistent measurement program. To define our measurement program fully, we included a specification of the GQM 'metrics' that identified the entity, attribute, unit scale type and counting rules, and the measurement aspects implied by the GQM questions.
- GQM needs to be supported by appropriate validation. We found an independent review by someone with metrics and data analysis experience a useful means of validating our analysis. The full specification of the GQM 'metrics' assists the review process by making assumptions about the GQM explicit rather than implicit.
- Avoiding the use of subjective measures was unanimously agreed as a method of ensuring that our data were reliable and comparable. It should however be noted that this decision was not readily accepted by project managers. We believe that project managers were reluctant to lose the element of discretion in project assessment. For instance, most people accept that development team characteristics (experience, knowledge on application domain, used techniques/tools, etc.) have a significant effect on productivity and defect rates. However, most project managers were extremely unhappy about substituting their personal subjective assessment of such characteristics with an objective measure based on the history of past projects in which the people were involved.
- In addition to the introduction of new measures, a systematic approach to establishing a measurement program should encourage the effective use of existing data.
- Measures can be collected in a consistent manner only if the data collection activities obey a defined company-wide procedure. The level of detail (maturity) of the software development process constrains the granularity of the measures.
References


