RT
219

Software Metrics Programme for Lebanese Software Companies

By
Rima S. Slim
B.Sc., American University of Beirut

PROJECT

Submitted in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science
at the Lebanese American University
February 1997

[redacted]

Dr. Nashat Mansour (Advisor)
Assistant Professor of Computer Science
Lebanese American University

5197

[redacted]

Dr. Ali Ataya
Professor of Computer Science
Lebanese University

Abstract

We introduce the subject of software measurement to software practitioners in Lebanon, and we propose a comprehensive process-oriented approach for an effective use of software metrics. We therefore overview related areas, for instance the Goal-Question-Metric paradigm and the Capability Maturity Model. We also present a set of software metrics. To be of practical use, we propose an exhaustive Software Metrics Programme to guide a typical Lebanese software company through the initiation and implementation of software metrics in its environment. We finally present a case study of the implementation of such a metrics initiative in a Lebanese company.

Acknowledgments

I greatly thank Dr. Nashat Mansour and Dr. Ali Ataya for their help and their effort. I also thank my colleagues for giving me valuable feedback. I can not close without thanking Dr. Wadih Jureidini for his time, and his concern.

Table of Contents

List of Tables

Chapter 1

Introduction

Controlling software production has been a permanent concern to software practitioners, regardless of the innovations that have occurred in the field. In fact, the process of developing new software and maintaining old systems has in many cases been poorly implemented, resulting in cost and time overruns and squandered business opportunities. Few projects are finished on time, within budget and meeting all user requirements. Executives wonder why software development is so costly and takes so long. Missed sales opportunities and customer dissatisfaction have to be avoided. In many cases, the company that reaches the market first with a new quality product is the company that gains market share over its competitors. This applies to software development as well. The software engineering industry is maturing. Customers are no longer willing to accept poor quality and late deliveries. Management is no longer willing to pour money into the black hole of IT (Information Technology), and more management control is now a key business requirement. Competition is growing. The software problem is as much a management problem as a technical problem. Improving the software products, the software process, and the productivity of development teams have become a primary priority to almost every organization involved with software production. Lebanese software companies are no exception.

Amidst this context, what motivates the subject of software measurement is the observation that we can not manage and control what we can not measure. Without measurement, it is impossible to detect problems during the software process before they get out of hand. Metrics can thus serve as an early warning system for potential future problems during a project.

Software metrics are defined as: *The continuous application of measurement-based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products.* [Goodman, 1993]

The benefits of incorporating software metrics within software development can be numerous. We would be in a better position to improve the software process and its products. We would enhance our ability to plan new projects, which leads to better business decision making. We will increase the confidence of our customers by demonstrating that we have a good knowledge of the strengths and weaknesses of our products and development process, and that positive actions are taken to correct the weaknesses. In short, metrication provides visibility to the entire software development process, hence reducing its apparent complexity.

Areas of application of software metrics are diverse. Some of these areas are cost and size estimation, prediction and assessment of quality levels for software, quantitative checks on software designs, comparison of one's organization with other organizations and provision of management information. So, software metrics provide information without which control of and improvement to the software production process would be performed randomly, non-systematically, and therefore with little efficiency. But we must realize that incorporating software metrics in the software production process means changing the way people work and think. Attempts to introduce the use of software metrics in any kind of ad-hoc way simply will not work. Software metrics initiatives fail because individuals who would never consider introducing a new computer system without a structured approach ignore the problems of introducing change inherent in software metrics implementation. Only by treating the implementation of software metrics as a project or programme in its own right with stages, plans, budgets, resources and management commitment can such an implementation succeed. Such a programme is referred to as a Software Metrics Programme, and its application in a company should be business driven, specific to the environment in question.

The objectives of this work are to introduce the subject of software metrics to Lebanese software companies, and to give such companies a Software Metrics Programme applicable to their size and nature.

We start in chapter 2 by an overview of areas related with software metrics, for instance the Goal-Question-Metric paradigm and the Capability Maturity Model. In chapter 3, we survey a set of software metrics. A Software Metrics Programme (SMP) that can guide a typical Lebanese company through the initiation and development of software metrics is presented in chapter 4. We apply this SMP to a Lebanese company, therefore providing a practical case study in chapter 5. In chapter 6, we conclude the report.

Chapter 2

GQM Paradigm and CMM

1. The Goal-Question-Metric Paradigm

The Goal-Question-Metric Paradigm, referred to as GQM paradigm, provides an informal mechanism for addressing the many kinds of problems which software practitioners think they might be able to solve through measurement [Basili and Rombach, 1988]. The idea is that any software measurement activity should be preceded by the identification of a software engineering goal which leads to questions, which in turn lead to actual metrics. Just what constitutes a goal or a question is left deliberately vague, and several levels of refinement may be required for certain goals and questions.

In other words, the GQM paradigm represents a systematic approach for setting goals, tailored to the specific needs of an organization, and defining them in an operational and tractable way. Goals are refined into a set of quantifiable questions that specify metrics. This paradigm also supports the analysis and integration of metrics in the context of the questions and the original goal. Feedback and learning are then performed.

The process of setting goals and refining them into quantifiable questions is complex and requires experience. In order to support this process, a set of templates for setting goals, and a set of guidelines for deriving questions and metrics have been developed.

These templates and guidelines are nonetheless not complete and will probably change over time with the growing experience in the subject.

Different sets of guidelines exist for defining product-related and process-related questions. Product-related questions are formulated for the purpose of defining the product, defining the product attributes of interest, and providing feedback relating to these attributes. Process-related questions are formulated for the purpose of defining the process, defining the relevant process attributes, and providing feedback relating to these attributes.

1.1. Templates for Goal Definition

Goals are defined in terms of purpose, perspective and environment.

1.1.1. Purpose

To 'characterize, evaluate, predict, motivate, etc.' the 'process, product, model, metric, etc.' in order to 'understand, assess, manage, improve, engineer, learn, etc.' it.

Example: To evaluate the system testing methodology in order to improve it.

1.1.2. Perspective

Examine the 'cost, effectiveness, correctness, defects, reliability, changes, product metrics, etc.' from the point of view of the 'developer, manager, customer, corporate perspective, etc.'.

Example: Examine the correctness of a product from the developer's point of view.

1.1.3. Environment

The environment consists of the following: process factors, people factors, problem factors, methods, tools, constraints, etc.

Example: The product is an operating system that must fit on a PC.

1.2. Guidelines for Product-Related Questions

For each product under study, three major issues need to be addressed.

1.2.1. Definition of the product,

1.2.2. Definition of the relevant attributes, and

1.2.3. Feedback relating to the attributes.

1.2.1. Definition of the product

The definition of the product includes questions related to the physical attributes of the product (such as size, complexity, etc.) and questions related to the cost of the product in terms of effort, computer time, etc. The definition also includes questions concerning changes and defects, that is a characterization of the errors, faults, failures, adaptations, and enhancements related to this product, and questions related to the general context, that is a characterization of the customer community using this product and their operational profiles.

1.2.2. Definition of the relevant attributes

The definition of the attributes of interest includes, for each attribute (e.g. reliability, user friendliness), questions related to the major models used and the validity of the model for the particular environment, that is an analysis of the appropriateness of the model for the specific project environment. The definition of the attributes of interest also includes the validity of the data collected, that is an analysis of the quality of data, the model effectiveness, that is a characterization of the quality of the results produced according to this model, and a substantiation of the model, that is a discussion of whether the results are reasonable from various perspectives.

1.2.3. Feedback

The feedback relating to the attributes includes questions related to improving the product relative to the attributes of interest, a characterization of the product quality, major problems regarding the attributes of interest, and suggestions for improvement during the ongoing project as well as during future projects.

1.3. Guidelines for Process-Related Questions

For each process under study, three major issues need to be addressed.

1.3.1. Definition of the process,

1.3.2. Definition of the relevant attributes, and

1.3.3. Feedback relating to the attributes.

1.3.1. Definition of the process

The definition of the process includes questions related to the quality of use of the process, and the domain of use of the process, that is a characterization of the object to which the process is applied to and an analysis of the process performer's knowledge concerning this object.

1.3.2. Definition of the relevant attributes

The definition of the relevant attributes follows a pattern similar to the corresponding product-oriented case, including, for each attribute (e.g. reduction of defects, cost effectiveness), questions related to the major models used, the validity of the model for the particular environment, the validity of the data collected, the model effectiveness and a substantiation of the model.

1.3.3. Feedback

The feedback relating to the process attributes follows a pattern similar to that of the product attributes.

1.4. Guidelines for Metrics, Data Collection, and Interpretations

The choice of metrics is determined by the quantifiable questions. The guidelines for questions acknowledge the need for generally more than one metric, for objective and subjective metrics and for associating interpretations with metrics. The actual GQM models generated from these templates and guidelines will differ from project to project and from organization to organization. This reflects their being tailored for the different needs in different projects and organizations. Depending on the type of each metric, the appropriate mechanisms for data collection and validation would then be chosen. The interpretation process should also be tailored to the specific needs of the environment in question [Basili and Rombach, 1988].

2. The Capability Maturity Model

The Capability Maturity Model (CMM) is a strategy for improving the software process, irrespective of the process model used [Schach, 1993]. The CMM was first put forward in 1986 by Watts Humphrey of the Software Engineering Institute (SEI) of Carneige Mellon University. Postulating that the cause of software problems is how we manage the software process, the belief underlying the CMM is that the use of new software methods will not in itself result in increased productivity and profitability. The CMM assists organizations by providing the infrastructure for achieving a disciplined and mature software process. This will lead to an improved process which in turn should result in improvements in the software quality. Software projects will therefore no longer suffer from time and cost overruns.

The software process incorporates both technical and managerial aspects of software production by comprising the activities, techniques, and tools used to produce software. The strategy of the CMM is to improve the management of the software process in the belief that improvements in techniques will follow as a natural consequence.

Since improvements in the software process can not occur overnight, the CMM induces changes incrementally. More specifically, five different levels of "maturity" are defined, and an organization then advances slowly in a series of small, evolutionary steps toward the higher levels of process maturity. The five maturity levels are the following, and are presented below.

2.1. The Initial Level,

2.2. The Repeatable Level,

2.3. The Defined Level,

2.4. The Managed Level, and

2.5. The Optimizing Level.

2.1. Maturity Level 1: Initial Level

This is the lowest maturity level. No sound software engineering management practices are in place in a level 1 organization. Instead, everything is done on an ad-hoc basis. If one specific project happens to be staffed by a competent manager and a good software development team, then that project can be a successful one. However, because of a lack of sound management in general and planning in particular, the usual pattern is time and cost overruns. Activities are almost always responses to crises rather than being planned tasks. In level 1 organizations the software process is unpredictable because it depends totally on the current staff. If the staff changes, so does the process. Within such an unpredictable process, it is impossible to foresee with any accuracy important items such as the time it will take to develop a product or the cost of that product. Unfortunately, the vast majority of software organizations all over the world are level 1 organizations.

2.2. Maturity Level 2: Repeatable Level

In level 2 organizations, basic software project management practices are in place. This level is the "repeatable" level because planning and management techniques are based on prior experience with similar products. At this level, *measurements* are taken, which constitutes an essential first step in achieving an adequate process. Typical measurements include the careful tracking of costs and schedules. At the repeatable level, problems are identified as they arise and corrective action is immediately taken before the problems get out of control. In contrast, a level 1 organization functions in crisis mode. *The key point is that without measurements, it is impossible to detect problems before they get out of hand.*

A level 2 organization also uses measurements taken during one project to draw up realistic time and cost schedules for future projects.

2.3. Maturity Level 3: Defined Level

At the third maturity level, the process for software production is fully documented. Both the managerial and technical aspects of the process are clearly defined. Continual efforts are made to improve the process in every possible sense. Reviews are used to achieve software quality. At this level, it is feasible to introduce new technology in order to increase quality and productivity further. In contrast, innovations only make the crisis-driven level 1 process even more chaotic.

A number of organizations have attained levels 2 and 3, but no organization has yet reached levels 4 and 5, although individual projects have. The two highest levels are thus aims for the future.

2.4. Maturity Level 4: Managed Level

A level 4 organization sets quality and productivity goals for each project. Quality and productivity are then continually *measured*, and corrective action is taken when there are unacceptable deviations from the goal. Statistical quality controls are used to enable management to distinguish a random deviation from a meaningful violation of quality or productivity standards.

2.5. Maturity Level 5: Optimizing Level

The goal of a level 5 organization is continuous process improvement. Statistical quality and process control techniques are used to guide the organization in this sense. The knowledge gained from each project is utilized in future projects. The process thus incorporates a positive feedback loop, resulting in a steady improvement in productivity and quality. The five maturity levels are summarized in Table 1.

Maturity Level	Characterization
1. Initial	Ad hoc process
2. Repeatable	Basic project management (measurement)
3. Defined	Process definition
4. Managed	Process measurement
5. Optimizing	Process control

Table 1. The five maturity levels of the Capability Maturity Model.

To improve its software process, an organization first attempts to gain an understanding of its current process. Then, it formulates the intended process. Actions that will result in achieving this process improvement are determined next, and ranked in priority order. A plan to accomplish this improvement is then drawn up and executed. This series of steps is repeated, with the organization successively improving its software process.

Experience with the CMM has shown that advancing a complete level usually takes from 18 months to 3 years, but moving from level 1 to level 2 sometimes takes 3 or even 5 years. This reflects how difficult it is to install a methodical approach in an organization that functions on a purely ad hoc and reactive basis.

For each maturity level, the SEI has highlighted a series of key process areas which an organization should target in its attempt to reach the next maturity level. Within each process area, there is a group of related goals which, if achieved, will result in the next maturity level being attained. For example, for maturity level 2, the key process areas include software quality assurance, configuration control and project planning. At the highest level, maturity level 5, the key practices include defect prevention, process change management and technology innovation. Comparing the goals of the two levels, it is clear that a level 5 organization is far in advance of one that is at level 2. For example, a level 2 organization is concerned with detecting and correcting faults through software quality assurance. In contrast, the process of a level 5 organization tries to ensure that there are no faults in the software in the first place, by incorporating defect prevention.

To aid an organization in its goal to reach the higher maturity levels, the SEI has developed a series of questionnaires which form the basis for an assessment by an SEI team. The purpose of the assessment is to highlight current shortcomings in the organization's software process, and to indicate ways in which the organization can improve its process.

The initial goal of the CMM program was to improve the software of the U.S. Department of Defense (DoD), but has then moved beyond this limited goal and is being implemented by a wide variety of different software organizations that wish to improve software quality and productivity.

Does implementing the CMM lead to increased profitability? Preliminary results indicate that this is indeed the case (e.g. at the Engineering Division of Hughes Aircraft in Fullerton, California) [Schach, 1993].

Chapter 3

Survey of Software Metrics

Three categories of metrics will be presented in this chapter:

1. Phase metrics, which are metrics specific to a certain phase of the software development process,

2. Quality metrics, which are metrics that deal with quality attributes of the software products and of the software process, and

3. Risk measures, which attempt to quantify the magnitude of risks using probability theory.

1. Phase Metrics

Phase metrics are metrics that are indicators only for a specific phase of the software development process. Assume a Waterfall Model, whose phases are the Requirements, Specification, Planning, Design, Implementation, Integration, Maintenance and Retirement phases. We will look at metrics relevant to the:

1.1. Requirements phase,

1.2. Specification phase,

1.3. Planning phase,

1.4. Design phase,

1.5. Implementation and Integration phases,

1.6. Operations mode, or maintenance, and

1.7. Metrics for testing, considering that testing is an activity that takes place all the way through software production and maintenance, rather than a separate phase.

1.1. Requirements Phase

1.1.a. Requirement specification change requests

Requirement specification change requests =

Number of change requests that are made to the functional requirements specification. [Paulish and Moller, 1993]

In case Rapid Prototyping is used during this phase, relevant metrics would be:

1.1.b. Measure of requirements volatility, i.e. how rapidly the rapid prototyping team determines the client's real needs. [Schach, 1993]

1.1.c. Measure of the number of requirements that change during the rest of the software development process. [Schach, 1993]

1.1.d. Measure of the number of times each feature in the rapid prototype is used when the clients and the users experiment with the rapid prototype. [Schach, 1993]

1.2. Specification Phase

1.2.a. A metric involved in measuring the size of the specification document is the measure of the number of pages in such a document. Such a measure allows to predict, by comparison, the effort that will be needed to build the product, bearing in mind the complexity of the specifications. [Schach, 1993]

1.2.b. A metric involved in measuring the quality of the specification document is the record of faults statistics found during specification inspections. [Schach, 1993]

1.2.c. De Marco's Bang metrics. [McDermid, 1991]

1.3. Planning Phase

Following are some size metrics needed for algorithmic cost estimation models.

- LOC, i.e. Lines of Code, [Schach, 1993]

- KDSI, i.e. Thousand Delivered Source Instructions, [Schach, 1993]

- FP, i.e. Function Points, [Schach, 1993]

- Measures provided by Software Science, [Schach, 1993]

- FFP metric, i.e. Files, Flows and Processes metric. [Schach, 1993]

1.3.a. Algorithmic cost estimation model (Model-based strategy)

COCOMO

In the Constructive Cost Model (or COCOMO), effort is estimated by:

Effort = a * size^b,

Where size is expressed in KDSI.

Three types of environments exist, the organic, embedded and semidetached environments, according to which the values of 'a' and 'b' are known. Cost drivers are later used to modify the basic result. [Schach, 1993]

1.3.b. Technique-based cost estimation strategies

Among the technique-based cost estimation strategies are:

- Expert judgment by analogy using the Delphi technique for reconciliation, [Schach, 1993]

- Bottom-up approach (Functional decomposition) [Schach, 1993], and

- The process of ranking functional entities. [Goodman, 1993]

1.4. Design Phase

Applied Design Metrics are a set of techniques, firmly based on measurement that are applied to designs at different levels to assess the complexity of those designs. Among the Applied Design Metrics (ADM) are McCabe metrics and Information Flow metrics. [Goodman, 1993]

1.4.a. McCabe metrics

McCabe metrics include both the Cyclomatic complexity and the Essential complexity.

a.1. Cyclomatic complexity

Applied against flowgraphs of designs.

McCabe cyclomatic complexity value is:

$v = e-n+2$, where

e = number of edges in the flowgraph, and

n = number of nodes in the flowgraph,

or, v = number of 'if statements' + 1. [Goodman, 1993]

a.2. Essential complexity

Is the measure of the structuredness of a design, and is applied on reduced flowgraphs.

McCabe essential complexity value is:

$vr = er-nr+2$, where

er = number of edges in the reduced flowgraph, and

nr = number of nodes in the reduced flowgraph. [Goodman, 1993]

1.4.b. Information Flow (IF) metrics

IF metrics model the degree of cohesion and coupling for a particular system component. Two models exist, the basic IF model and a more sophisticated IF model.

b.1. The basic IF model

IF metrics are applied to the components of a system design. Three measures are defined for a component 'A':

- FAN IN: number of other components that can call or pass control to 'A',

- FAN OUT: number of components called by 'A',

Then, IF index of 'A' is: IF(A)= (FAN IN(A) * FAN OUT(A))^2. [Goodman, 1993]

b.2. A more sophisticated IF model

- FAN IN = a+b+c+d, where

a = number of components that call 'A',

b= number of parameters passed to 'A' from components higher in hierarchy,

c= number of parameters passed to 'A' from components lower in hierarchy,

d= number of data elements read by 'A'.

- FAN OUT = e+f+g+h, where

e = number of components called by 'A',

f = number of parameters passed from 'A' to components higher in hierarchy,

g = number of parameters passed from 'A' to components lower in hierarchy,

h = number of data elements written to by 'A'.

Then, IF index of 'A' is: IF(A)= (FAN IN(A) * FAN OUT(A))^2. [Goodman, 1993]

Other than the ADM, design faults is a metric used during the design phase of the software development phase.

1.4.c. Design faults

This metric is defined as the count of faults found during all design reviews of a certain software design. [Paulish and Moller, 1993]

1.5. Implementation and Integration Phases

1.5.a. Code faults

This metric is defined as the number of faults found during all code reviews for the software product. [Paulish and Moller, 1993]

1.5.b. Productivity during implementation is measured as:

Productivity = LOC or FP / Effort measured as time. [Paulish and Moller, 1993]

1.5.c. From a testing viewpoint, relevant metrics include the total number of test cases and the number of test cases which resulted in a failure. Also, detailed statistics of the types of faults detected are useful for code inspections purposes. [Schach, 1993]

1.6. Maintenance Phase

1.6.a. Customer complaint rate

This metric is defined as the number of customer identified faults per time period from the time of first field use of the product through its lifetime. [Paulish and Moller, 1993]

1.6.b. Metrics applicable to activities pertaining to the development process and maintenance are applicable during this phase. [Schach, 1993]

1.6.c. Measures relating to software fault reports such as the total number of faults reported and the classification of those faults by severity and fault type, as well as information regarding the current status of the fault reports can be used. [Schach, 1993]

1.7. Metrics for the Testing Activity

1.7.a. Test faults rate

This metric is defined as the number of faults discovered per time within each process testing phase for each product for each release. [Paulish and Moller, 1993]

1.7.b. Test planning metrics such as the number of test cases planned, and the number of planned and unplanned test cases run. [McDermid, 1991]

1.7.c. Test coverage metrics, such as the percentage of statements and/or branches covered by a set of test cases. [McDermid, 1991]

Metrics for inspections measure the effectiveness of the inspection process.

1.7.d. Fault density: Is the number of faults per page for specification and design inspections, or the number of faults per KLOC (or other) for code inspections. [Schach, 1993]

1.7.e. Fault detection rate: Is the number of major/minor faults detected per hour. [Schach, 1993]

1.7.f. Fault detection efficiency: Is the number of major/minor faults detected per person-hour. [Schach, 1993]

2. Quality Metrics

Quality is the satisfaction of user or customer requirements. Some product quality attributes are reliability, usability, maintainability, enhanceability and correctness.

2.1. Reliability

Is the ability of a software product not to fail. Relevant metrics are:

2.1.a. Defect density measure

Defect density measure =

Defects reported from the customer field / Size of the product. [Gillies, 1992]

2.1.b. MTTF, Mean Time To Failure

This is a measure of the time between observed failures.

MTTF = t_{tot} / n_t, where

t_{tot} = total time period of use, and

n_t = number of failures in t_{tot}. [Gillies, 1992]

2.1.c. Complexity measures (McCabe's etc.) can be used as a measure of reliability, under the assumption that as complexity increases, reliability decreases. [Gillies, 1992]

2.1.d. Probability of failure on demand

This is a measure of the likelihood that the system will behave in an unexpected way when some demand is made on it. It is most relevant in safety-critical systems and 'non-stop' systems whose continuous operation is critical. [Sommerville, 1992]

2.1.e. Rate of failure occurrence (ROCOF)

This is a measure of the frequency of occurrence with which unexpected behavior is likely to be observed. For example, if the ROCOF is n/100 this indicates that n failures are likely to occur in each 100 operational time units. [Sommerville, 1992]

2.1.f. Availability

This is a measure of how likely the system is to be available for use. [Sommerville, 1992]

2.1g. System test faults

System test faults =

Total number of software faults reported by the testing function during system test / Size of the product. [Paulish and Moller, 1993]

2.1.h. Customer change requests

Customer change requests =

Number of unique change requests made by customers during a certain period of field use of a software product release / Size of that release. [Paulish and Moller, 1993]

2.2. Usability

Is the ease of use of the software product. Relevant metrics are:

2.2.a. Number of user complaints, defect, fault, or error reports from the customer field, different than those relating to reliability, per module. [Gillies, 1992]

2.2.b. Readability of the software product's documentation is used to assess how such a documentation assists in the usability of the software product. Two methods that can be used are the Flesh-Kincaid Readability Index, and the Fog index. [Gilllies, 1992]

2.3. Maintainability

Is the effort required to locate and fix a fault in the program within its operating environment. Relevant metrics are:

2.3.a. Corrective maintenance.

Let t = time it takes to fix a defect,

d = number of defects fixed, and

n = number of defects outstanding,

then, for a given time period,

Maintainability = $t * (d+n)$. [Goodman, 1993]

2.3.b. Complexity measures (McCabe's etc.) can be used as a measure of maintainability, under the assumption that as complexity increases, maintainability decreases. [Gillies, 1992]

2.3.c. Readability of code can be used as a measure of maintainability. De Young and Kampen measure readability in terms of the number of statement lines, the average length of variable names and the total number of program branches. [Gillies, 1992]

2.3.d. Modularity can be used as a measure of maintainability, since increased modularity is generally assumed to increase maintainability. [Gillies, 1992]

2.3.e. Testability can be used as a measure of maintainability, since the ease and effectiveness of testing will have an impact upon the maintainability of a software product. [Gillies, 1992]

2.3.f. Mean Time To Repair (MTTR). The probability that when maintenance action is initiated under stated conditions, a failed system will be restarted to operable conditions within a specified time. [Gilb, 1988]

2.4. Enhanceability

Is the ease by which requested changes can be made to a system. A relevant metric is:

2.4.a. Enhanceability metric

Enhanceability =

Number of components affected by a change / Total number of components in system. [Goodman, 1993]

2.5. Correctness

Is the extent to which a software product fulfills its specification. Relevant metrics are:

2.5.a. Error prediction can be used as a measure of correctness. Ottenstein predicts the number of errors to be found during validation and the total number of errors found during development. This measure is based upon the assumption of a stable software development environment. [Gillies, 1992]

2.5.b. Error detection can be used as a measure of correctness. Remus and Zilles' model of defect removal uses the number of detected errors and a measure of error detection efficiency to predict the number of errors remaining undetected. It is assumed that the number of defects is proportional to the length of the program and that the rate of defect removal can be predicted. [Gillies, 1992]

2.6. Process Quality

Monitoring process quality involves both schedule and productivity metrics.

2.6.a. Schedule

Ability to manage a software project on schedule.

Schedule =

(Planned work time to achieve a certain milestone - Actual work time to achieve this milestone) /

Planned work time. [Paulish and Moller, 1993]

2.6.b. Productivity

Productivity is generally defined as:

Productivity = Work product / Cost to produce that product.

Generally speaking, productivity within software development is expressed as:

Productivity = Project size / Project effort. [Paulish and Moller, 1993]

3. Risk Measures

Risk-oriented metrics attempt to quantify the magnitude of risks using probability theory. A variety of techniques are employed but most involve establishing probabilities on the basis of historical data, mathematical models and human expertise. [Rae and Robert, 1995]

3.1. Risk Exposure

Boehm uses probabilities to calculate a monetary value for Risk Exposure (RE), which is the product of the potential loss associated with a given risk and its probability of occurrence. Risk exposure thus focuses attention on risk items which have both a high probability of occurrence and a high potential for financial loss. Risk Exposure may be expressed mathematically as:

Risk Exposure (RE) = Prob (UO) * Loss (UO),

where UO stands for Unsatisfactory Outcome. [Rae and Robert, 1995]

3.2. Risk Reduction Leverage

Boehm also proposes a measure for Risk Reduction Leverage (RRL), allowing evaluation of competing risk reduction strategies. Risk reduction leverage is found by calculating the potential reduction in risk exposure afforded by a particular risk reduction strategy divided by the cost of implementing this strategy. The bigger the number, the more effective the strategy. Risk Reduction Leverage may be expressed mathematically as:

Risk Reduction Leverage (RRL) = (RE_{before} - RE_{after}) / Risk reduction cost,

where RE is the previously defined Risk Exposure. [Rae and Robert, 1995]

Chapter 4

Software Metrics Programme

The Software Metrics Programme (SMP) proposed consists of the following five stages that will be presented in detail in this chapter.

1. The Initiation Stage, during which the need for measurement is noted in the organization, responsibilities are assigned and the feasibility of the measurement initiative is assessed.
2. The Requirements Definition Stage, during which we aim at collecting the various measurement requirements of the various groups in the organization.
3. The Component Design Stage, during which we aim at designing a global measurement solution of the SMP.
4. The Component Build Stage, during which we aim at building the various components of the previously designed measurement solution.
5. The Implementation Stage, which consists of the continuous implementation of the measurement initiative in the organization. [Goodman, 1993]

1. The Initiation Stage

The initiation stage is the first stage of the Software Metrics Programme, hereafter referred to as SMP, or programme. Decisions taken during this stage set the scene for the rest of the programme. In fact, it is during this stage that the first contact is established with the customers. In this context, the customers are the customers of the SMP, that is the software organization in question that wishes to introduce measurement to its environment. So, the orientation is on the customer angle. The initiation stage consists of three main activities:

- 1.1. Having an initial management decision,
- 1.2. Assigning management responsibility, and
- 1.3. Performing a feasibility study and presenting the feasibility report to management.

1.1. The initial management decision

Two factors lie behind the start of an SMP in a software organization: an organizational trigger and an initiator. The trigger comes from some need within the organization, and the initiator realizes that the need exists and seeks to satisfy it. Generally, an initiator identifies a need for measurement, determines that something will be done to satisfy that need, but may delegate the responsibility for the development and implementation of the measurement initiative to other persons at more tactical and operational levels. Initiators set out the scope of the work and are the ones who should obtain and give the initial authorization to proceed with the programme. Recognizing the triggers is important because they affect the SMP by driving it down a particular road, thus giving the initial focus. However, the scope of the programme can be later altered and broadened during requirements gathering. Usually, multiple triggers will be present. This tends to widen the scope of the work, so that difficulties, and even failures in one area, can be outweighed by successes in others. But widening the scope of the work can also slow things down. On another hand, a balance between constraining or tightly-defined requirements and a vague problem definition that is liable to drastic changes should be reached. As a general rule, it is better to tend towards vagueness rather than rigor at that stage. All through, it should be made very clear that the SMP is not a quick fix because it is about fixing a process, not a single fault. We present below a set of different situations or events that can trigger an SMP within an organization:

- General quality or productivity concerns,
- Concern about customer complaints,
- Customers' requirements for increased information,
- Knowledge of competitors' activity in the metrics area,
- Size, time and cost estimation concerns,
- Quality programmes,
- Management pressure for more information,
- A consultancy recommendation, and
- New awareness of the possibility of measurement (perhaps as a result of someone attending a seminar or a conference).

To sum up, both the triggers and the initiator are identified at that point of time. The decision to do something about software metrics in the organization is taken during this first step of the initiation stage.

1.2. Assigning management responsibility

Somebody has to take responsibility for the work being done. So, a relatively senior involved person within the organization is sought to be appointed as the sponsor of the SMP. He or she will act as the final arbiter for the measurement initiative as to decisions. The sponsor is usually not from the team doing the actual work, but someone with authority within the organization and with access to senior management. This is because the proposal for further work at the end of the initiation stage will need to be authorized. The sponsor is also independent of the other functions but should not lack credibility because of that.

Some simple guidelines for the sponsor's selection are that the more senior and involved this person is, the better, and that he or she should not change over the lifetime of the measurement initiative.

1.3. Performing a feasibility study and presenting the feasibility report to management

A feasibility study is then performed to see whether the measurement initiative is feasible within the organization. Such a study is also performed to convince the organization that the effort will pay back the cost, i.e. the measurement initiative is cost-effective, and to convince senior managers in doubt that it is possible to control the software development process through measurement. So, a few preferably full-time persons (from two to four) will be assigned to that task, therefore forming a feasibility team that reports to the sponsor of the SMP. Such persons should know the business, and should be effective interviewers to be able to get valid and useful information from other people. Good candidates for such a job are therefore good systems analysts. However, to overcome the lack of metrics experts, a software organization could look for the help of consultants.

To perform such a task, the feasibility team needs to perform an initial subject familiarization with the metrics area and an initial market research, as described below. Actually, the organization will only be fully convinced of the benefits of software metrics by demonstrating the successful potential use of the metrics within the organization itself. This is done by showing that the way in which others have successfully used metrics is applicable to their own organization. Three tasks should be done in this sense: the feasibility team should be aware of the work of others, they should establish their own needs in the organization and they should try to map the work of others to their own needs and present this mapping. Talking to other practitioners, to academics and to consultants would help in this track.

In addition to the initial subject familiarization, an initial market research is performed. The market in this context is the market formed by the different persons in the organization in question involved with the measurement initiative. The feasibility team has to establish the needs of such persons. To identify the customers, the organization's staff is divided into a number of functional groups (e.g. managers, software developers, testers, designers, etc.). All SMP customers or a sample from each group are selected for an interview, but anyway, the most receptive and the less positive customers should be accounted for as well. The interview should address certain areas like introducing the subject of software metrics, ascertaining the problems faced by the interviewees during their work and trying to know what additional information can be beneficial to the project. The results of the interviews are then documented and stored in the SMP repository. Out of such information, the feasibility report is prepared. Finally, in order to get authority to proceed, the feasibility team gets access to the most senior manager possible and presents this feasibility report.

So, the initial scope of the SMP is set during this first stage. The best advice to give is to listen carefully and constructively to all parties involved.

2. The Requirements Definition Stage

Having obtained the authority to develop the programme during the initiation stage, this stage consists of several activities:

- 2.1. Initial publicity campaign of the SMP within the organization,
- 2.2. Establish interface through the formation of the Metrics Coordination Group (MCG),
- 2.3. Identify potential super champions,
- 2.4. Obtain specific requirements that will drive the SMP,
- 2.5. Establish initial definitions,
- 2.6. Prioritize the various collected requirements, and
- 2.7. Consolidate the requirements to produce a consolidated requirements specification.

2.1. Initial publicity campaign of the SMP within the organization

An initial publicity campaign of the SMP within the software organization in question is performed as the first step of the requirements definition stage. The metrics team, composed of the measurement consultant, the sponsor and the feasibility team will try to "sell" (or to market) the SMP. Marketing the SMP is necessary for the future cooperation of the staff with the programme, which is vital to the requirements capture, to the feedback needed from the staff, etc. So, everybody should be informed that an SMP now exists in the organization. The metrics team should convince people that their cooperation is worthwhile, but at the same time avoid overselling the programme by making exaggerated claims and building up false hopes. The aim is changing the way people work and think, so the metrics team should start a continuous education process within software development. The use of in-house magazines or articles, site notice boards, or open briefing sessions is useful in this campaign. A software metrics workshop can also be used both to stimulate interest in the SMP and to provide a vehicle for education, during which outstanding staff members are identified. Two issues should be emphasized. First, the metrics team members should not loose track of the aim they had set to such a campaign, which is usually to involve others with the measurement initiative. Second, they should account for the effort, time and budget that such a campaign needs.

2.2. Establish interface through the formation of the Metrics Coordination Group (MCG)

The interface between the metrics team and the rest of the staff within the organization is established through the formation of the Metrics Coordination Group (or MCG). The MCG acts as a steering committee, or board of directors of the SMP. This interface is needed especially if the staff is large. The MCG should contain one or more representatives from each customer group (preferably senior persons), the metrics team, the sponsor and the measurement consultant. The MCG should ensure that the SMP addresses the business needs of the organization. In this direction, it has to demonstrate that the users have been consulted from the start so that a sense of ownership is achieved. The MCG should also act as a final review body for any product from the SMP. An SMP product labeled "Approved by MCG" would have added credibility. MCG members also assess in periodical meetings the course of the SMP, and inform others about these issues.

In a first meeting, the MCG would use information from the feasibility study to get more acquainted with measurement issues. So, the MCG has to be guiding the direction of the Software Metrics Programme. Its members should be thanked publicly and often. At that point of time, the true launch of the SMP is achieved.

2.3. Identify potential super champions

An equally important step in the requirements definition stage is to identify potential super champions among the staff of the software organization in question. Such persons distinguish themselves from the others by their added interest, involvement, and creativity as to measurement issues. They adopt the measurement initiative, become a part of it, and even drive it further. They become centers of expertise, enthusiasm and excellence, vital to the success of the initiative.

2.4. Obtain specific requirements that will drive the SMP

This is the core activity of the requirements specification stage. Obtaining the set of specific requirements that will drive the SMP is not a task that can be easily defined, because it includes interaction with people. Essentially, there exists two techniques that can be used to capture the specific requirements within the SMP, a systems analysis interview-based approach and a workshop-oriented brainstorming approach.

2.4.a. Systems analysis interview-based approach

In this first approach, capturing and specifying the requirements of the SMP is done with the same procedures as classic systems analysis, i.e. interviews. Here also, the foundation of good systems analysis is preparation.

So, in the systems analysis interview-based approach, the measurement initiative consultant and sponsor, with the help of the metrics team, identify specific key players within the customer set. Structured interviews are then carried out with these individuals.

A two-pass interview approach can be used. For each of the players, the first interview aims to identify their particular role within the organization as they see it, what processes they are personally involved in and what those processes entail, what responsibilities, in terms of deliverables, they have and, most importantly, what they perceive to be the weaknesses in the processes they operate. Second, the metrics team will help the interviewees formulate their requirements from the measurement initiative. Talking to people takes time, and to get the most out of the available time, there will need to be a plan and structure to the interview sessions.

Taking requirements from a number of sources will lead to similar requirements being identified, such that the focus would be on requirements with general need. So, requirements should be grouped into classes. The following list gives one possible set of headings that can be used to group metrics requirements and that can also be employed to focus on interview and workshop sessions:

- project metrics,
- process metrics,
- project control,
- cost estimation,
- quality assessment and prediction, etc.

Some guidelines to follow are not to schedule more than three to four sessions per day, and not to start at the top because the team performing the interviews is probably not yet prepared for top management. The plan should be slightly modified when interviewing senior managers because of time concerns. Out of business etiquette, and to keep the SMP "popular" in the organization, the metrics team will ask for a manager's permission when it intends to interview one of its staff.

2.4.b. Brainstorming

The second way of capturing the SMP's requirements in a software organization is brainstorming. This approach centers on a workshop and involves essentially the measurement consultant, the sponsor, and the metrics team simultaneously working out a set of requirements for the SMP with the organization's staff. This approach requires a fair amount of planning and coordination and also a strong meeting leader, to control events as they occur. On the positive side, it can give a set of agreed requirements very quickly, namely, at the end of the workshop. This approach presents however greater risk than the first one, because it does not involve all key players individually.

Any material that can be used to introduce the topic of software metrics should be prepared for use in such a workshop. The team responsible for the workshop will have to define software metrics, to introduce benefits of measurement and to relate these benefits to the software development environment. The team's members will also show examples of the effective use of measurement and relate the information shown to the identified business needs of the organization. The examples are mainly gathered from other practitioners, from academics involved in the metrics area, from measurement consultants, or simply from the literature.

Once the team has this introductory material, it will need to identify the key players, from within the staff of the organization, that it intends to use to define the specific requirements. Also it has to arrange some form of management sponsorship to ensure that it will get the people it wants to the workshop.

Afterwards, the team should plan out the structure of the session in some detail to ensure that things do not get out of control and that the required information will actually be gathered during the workshop. Usually, a workshop will take one full day. The team will explain that the purpose of the workshop is to derive a specific set of requirements that will be used to drive the SMP in the organization. It will then present the previously prepared introductory material, after which 'Questions and Answers' discussions are handled. The team should steer discussions towards requirements definition in the afternoon, in order not to run out of time. To focus on requirements capture, attendants can be asked if they would like to try any of the approaches outlined in the examples, what kind of information they would like to have that they do not get now, etc. So, the metrics team will have to propose measurement requirements to the attendants in order to help them identify and formulate their requirements of the SMP.

Whatever approach is used, the team should end up with a set of specific requirements that have come from the business and that the business wishes the SMP to address. These requirements will fall into a number of categories, those with quick pay back and to which there are reasonable answers, those classified as research issues, and obtuse requirements. These multiple requirements may have to be prioritized if they can not be addressed all at once. The issue of prioritization is addressed later in this stage.

One other thing that may well be discovered is the need to establish some initial definitions, and this is addressed in the next step of this stage.

2.5. Establish initial definitions

Establishing initial definitions is left until now to identify the terms that can cause problems, through talking to the different parties in the organization. Since terminology can be one big barrier to understanding, these basic terms should be defined, to leave no room for interpretation and modification. These terms will have to be defined in a meaningful way, acceptable to the organization, and not as theoretical abstract concepts. The necessary agreement to these definitions is obtained by presenting them to the MCG for approval. Anyway, the principle of pragmatism and compromise should be applied. Different people will use these terms and a unique consensus can probably not be reached.

Another issue that should be settled is the adoption or derivation of a high-level development process model for the organization.

2.6. Prioritize the various collected requirements

The various requirements obtained for the measurement initiative may be hard to address all in one time, and therefore may have to be prioritized. Prioritization of these collected requirements will be done essentially by how easy they will be to satisfy. So, two tasks will have to be performed; first, identifying the available data sources and relating the types of available information to the identified requirements, and second identifying storage, analysis and feedback requirements.

To identify available data sources, the metrics team needs to look at what is currently available in terms of data and where this data resides. In order to do so, it has to talk to the people in the organization. The MCG can help in this sense. Certain data collection mechanisms may already be in place within the organization, still the validity of the data on hand should be checked. This is especially true since there could be a lack of care in recording information, which seems to be a direct result of the lack of feedback from these established systems. The information database on hand may well be a "write-only" database. The metrics team should get people to ask "Why are we collecting data?" rather than "What data should we collect?". It can change existing systems or put new systems in place, accounting for the cost involved in each of its decisions.

At this point, after noting the type of information that is available, the metrics team should relate this information to the various identified requirements of the SMP. This will give an indication of what is involved in satisfying those requirements and will help prioritization.

The second task to perform in order to do the prioritization of the collected requirements is to identify storage, analysis and feedback requirements. Almost all requirements associated with an SMP imply the need to store, analyze, and feedback information. Taking a look at the scale of these tasks helps in prioritization. The metrics team will have to ask questions like: "How much data will we have to collect and from how many sources?", "Are we talking about a straightforward analysis or a complex one?", and the team has to remember that in the latter case, it will need a great deal of data that will take time to collect. It also has to consider: "How many customers need to have information fed back, in what form and at what cost?". For example, if the team members are handling senior management reporting, then they will need to spend some time ensuring that they present their results in a business-like and professional way.

Some people will argue that such tasks belong to the design stage, but the metrics team needs to get some idea of what is involved in satisfying a requirement to be able to decide whether this requirement will be satisfied immediately.

2.7. Consolidate the requirements to produce a consolidated requirements specification

Having got a good idea of what the SMP's requirements are, and what they may involve, the metrics team has to put it all together into a consolidated requirements specification. A good way of doing that is to group various identified requirements into goals. Two examples are given below.

Example 1

The metrics team may have the following set of identified requirements linked to specific customer groups:

♦ Improve cost estimation techniques for enhancement projects - Product development team leaders.

♦ Improve cost estimates that feed into the planning process - Planning manager.

♦ Sort out estimates - Systems development director.

The goal or consolidated requirement becomes:

Improve the cost estimation process.

Example 2

Alternatively, the metrics team may have identified requirements linked to specific customer groups that go something like:

♦ We are discovering faults too late, we need to sort them out earlier - Software development manager.

♦ Our inspections are a joke. We need some objective criteria that will get rid of the subjectivity in deciding when a design is good enough - Senior software engineer.

♦ Why don't we use Information Flow metrics as we were taught at university? - Junior software engineer.

The goal or consolidated requirement becomes:

Identify and implement non-subjective techniques that can identify error-prone components at the inspection stage, or earlier, of the design phase.

A typical, non-exhaustive set of consolidated requirements, expressed very loosely, is presented below:

- Improve cost and size estimation,
- Improve project control procedures,
- Provide management information about performance, covering productivity and achieved quality,
- Address the prediction of quality attributes prior to release, and
- Address the assessment of designs and requirements.

The metrics team should link each consolidated requirement to a set of primary and secondary customers. The primary customer will effectively be the individual, or functional group, that will decide whether or not the metrics team has satisfied the requirement for measurement. The secondary customers are those who have a view of what the team delivers, and will be expected to provide information to help satisfy the requirement.

Having consolidated the requirements, the metrics team presents these consolidated requirements to the measurement sponsor at a minimum for a review. It is also a good idea to present them to the MCG, so that the organization validates the work to be done.

So, during this requirements specification stage, the metrics team found out what the various parts of the organization want from the SMP by capturing the requirements. These requirements are communicated back to the organization, so that agreement about the objectives of the programme is gained.

The SMP team members spent a considerable amount of time and effort ensuring that they know, and that the organization agrees that they know what is required from the SMP. Now it is time to satisfy these requirements.

3. The Component Design Stage

After the initiation stage and the requirements definition stage, the metrics team members know a lot more about software metrics than they did when they first started working on the measurement programme. They have identified a set of requirements against the programme linked to customer sets within the organization and they have an idea about what information the organization already collects. They have also established a steering group, the Metrics Coordination Group (MCG).

The Component Design stage consists of four main streams or sets of activities that will eventually come together and result in a design proposal for the measurement initiative. These four streams are:

- 3.1. The Metrics Definition Stream,
- 3.2. The Administration Design Stream,
- 3.3. The Marketing and Business Planning Stream, and
- 3.4. The Infrastructure Design Stream.

Each of these streams with its related activities is presented in detail below.

3.1. The Metrics Definition Stream

The tasks performed during this stream constitute the core of the SMP. In fact, the definition of the metrics and the metrics-based techniques and tools is performed at that point of time, after all the preliminary preparations are done. The metrics definition stream consists of two activities:

3.1.a. Model definition through the Goal-Question-Metric paradigm, and

3.1.b. Identification of external products.

3.1.a. Model definition: Goal-Question-Metric (GQM) paradigm

The identified requirements related to customer groups could range from generalities to very specific requirements. The metrics team has, in terms of Basili and Rombach's GQM paradigm (refer to Chapter 2) the goals defined, and the means by which the questions can be resolved. The goals are the measurement requirements collected, and should be formulated according to the templates for goals definition presented in Chapter 2. The metrics team should ensure that these goals are actually measurement goals. Individuals will often throw a general goal that is not suitable as a requirement within a metrics programme. For example, a manager may claim that his goal is to increase development productivity by 25 percent within 2 years. This is a valid strategic goal but it does not help a metrics programme. Acceptable measurement requirements within this manager's strategic goal may include "provide a mechanism by which the productivity levels can be assessed in order to be improved".

In the consolidation of requirements step of the previous stage, requirements classification has been done. Similarities between the different requirements have been noticed leading to the classification of these requirements into a small number of major categories or macro-requirements. One advantage of this grouping is to enable the satisfaction of as many individual requirements as possible in the shortest time possible, since the number of customers and individual requirements exceeds the available resources if the requirements are to be addressed individually. Then, generic solutions to these requirements are offered that would be tailored to meet the needs of specific customers. One other advantage that comes from requirements grouping is that the metrics team can more easily use other people's solutions by identifying external solutions and applying tailoring to use in the organization (for example in the case of cost estimation packages that run on a PC).

Each customer group within the organization who is linked to a measurement requirement then appoints a person who is expected to express the needs of the group, and to represent its interests during the process of satisfying the measurement requirements. This role is important especially if the groups are large. The ideal candidate to hold such a role is the primary customer for the specific requirement solution.

Having carried out this requirements grouping, the metrics team starts to go through the question or modeling task of metrics definition. Almost all requirements laid against an SMP relate to attributes of either the development process or its products. That is the requirement should be categorized as a process-oriented or a product-oriented measurement requirement, by identifying the attribute of interest within it. Then, for any attribute within a measurement requirement, three issues need to be considered.

♦ First, the metrics team should define the attribute, i.e. what is meant by that attribute. One set of definitions is found in ISO IS9126. The initial definitions set previously help at that point.

♦ Second, it should be realized that there are two areas of interest for any attribute. The attribute can be monitored (or assessed), alternatively prediction can be applied to the attribute. The metrics team should treat these two areas as separate requirements within the SMP.

♦ Third, the metrics team should derive a model that will satisfy the measurement requirement. Therefore it should identify the characteristics of the process or of the product that affect the attribute of interest. This can be performed by asking people best placed to answer the question "What affects this attribute?". More definitions can be required. Having this set of characteristics, the team starts to combine these in the form of a model. In this context, a model is a mathematical combination of characteristics resulting in a function for which the dependent variable provides information about the attribute of interest, along with the technique built around the analysis of that function.

Finally, it should be noted that modeling is a human activity, and will most probably contain an element of subjectivity and an element of simplification. There is no "perfect" model. Simpler models are probably better than more complicated ones. Some of the characteristics identified may be ignored. Models can be changed or incrementally developed and extended.

Now, the team should derive metrics from the model. The model, or at least the component of the model that is a mathematical function, gives the *composite metric* that will be used to measure a particular attribute. By looking at the components of the composite metric, the team will identify the *base metrics*, that is the raw data that will be collected in order to provide information about a particular attribute. The metrics team should define the base metrics in a meaningful way, such that their collection is feasible within the organization.

An important alternative to internal modeling that can often be used is to find out what is the "state of the art" or the "accepted wisdom" relating to the metrics requirements on hand. To do that, the metrics team looks at what is being practiced in other organizations and at what is offered in the literature. By talking to people in the organization, the metrics team makes sure that what they propose is practical.

It should also be noted that metrics derived are not and can not be perfect. They should satisfy the requirements in a pragmatic way, especially that some of the characteristics identified as affecting the attribute in question may be ignored for some reason. Therefore, customer groups concerned with measurement should be satisfied with coarser measures with reasonable indication of the attribute. Accounting for the customer sets, listening to valid criticisms, and changing the model and the metrics if necessary are guidelines to follow. The implementation of the defined composite and base metrics is then performed.

So, to summarize the steps in the metrics derivation, relating that to the GQM paradigm, the steps are as follows.

Goals

Identify and refine the initial measurement requirements and their associated customer sets.

Questions

Identify the attribute that each measurement requirement addresses. Define the attribute. Identify characteristics of the product or process that affect the attribute of interest. Derive a model by combining characteristics, or adopt a known solution after finding out what others are doing.

Metrics

If modeling is used, derive the composite and base metrics from the model.

3.1.b. Identification of external products

In some situations, there are already proprietary products, tools or techniques available on the "market" that the metrics team can directly adopt, thus circumventing the derivation of metrics process. The team will then move on to implementing the use of those solutions directly. This process can be viewed as an "import" of products or techniques. Nonetheless, some work will still be required to adapt these solutions to the organization's own environment.

The "goals" step is a before. It is in the "questions" area that the most significant difference is seen. The metrics team in this case investigates the market place to see what is on offer, decides which of the available tools is best suited to the measurement needs, given the specific requirements and constraints. Hands-on experience is recommended to see the ease of use and suitability of such products (for example, cost estimation tools). The tools will tell what information they require as input and these are obviously the base metrics to collect. The implementation of metrics can then be performed.

The derivation model can also be modified to apply it to the selection of "public domain" metrics, like variants of know metrics (for example, FPA, McCabe's and Information Flow metrics), if these are relevant to the measurement requirements collected. Many of these products come complete with analysis techniques or guidelines for use.

A good way of researching the market to see what is offered is to attend training courses and metrics conferences.

3.2. The Administration Design Stream

After defining the set of metrics that answer the collected measurement requirements, the metrics team should consider areas of collection, storage, analysis and feedback during this second stream of the Component design stage.

This stream covers elements of the SMP that are needed to use the metrics in a practical way. It involves designing the operational procedures and the support mechanisms necessary for the implementation of the measurement initiative. This stream consists of four main tasks:

3.2.a. Mapping base metrics to available data,

3.2.b. Establishing links to data administrators,

3.2.c. Defining data collection mechanisms, and

3.2.d. Designing storage, analysis and feedback mechanisms.

3.2.a. Mapping base metrics to available data

After the base metrics are defined, the raw data they represent needs to be collected. Data that is already collected by the organization in question should be used if possible. This fact presents two benefits. First, it reduces the amount of work to be done, since no new collection systems have to be designed. Second, it reduces the amount of additional work that the data suppliers, mainly the software engineers, have to do. But there are also some problems in using existing data collection mechanisms. In fact, the data supplied will not be exactly what is wanted for the SMP, and such systems are often viewed negatively by the staff as hindering real work, and may therefore provide incorrect data. If the problem with an existing collection system is that the specific data that the metrics team requires is not there, then the obvious answer is to change the system so that it will supply the required item. Another option is to change the requirement on the existing system. Often there is another data item available that can almost give the metrics team what it wants, or, alternatively, there is another method by which the team can obtain the data. The metrics team may have to go as far as changing the composite metric if the raw data is not available and can not be collected fairly easily. If the data is being supplied, but is inaccurate, then a more fundamental problem exists and will have to be dealt with. In any way, the metrics team should try to reach an optimal mapping between the data that is already available in the organization and the defined metrics.

3.2.b. Establishing links to data administrators

Once the metrics team members know in detail the kind of information that is available in the organization and that they will use in the measurement initiative, they should form a link to the people who are responsible for administering the data collection systems that exist within the organization. Such links will most probably be already present. Now is the time to formalize them under management sponsorship. The metrics team members will document their requirements from the data administrators, and have these latter formally agree to those requirements. Data administrators will usually have one of two extreme reactions. On the one hand, they will see the metrics team members as a major inconvenience, on the other hand, they will appreciate the fact that the data that they collect will be actually used. In any case, data administrators are to be thanked publicly, and often.

3.2.c. Defining data collection mechanisms

Before going live with the measurement programme, the metrics team members should try to have as many of the bases covered as possible. So, they will help the software developers in "how" to supply data. They will define forms to be filled, decide on how often the developers should forward information, and teach them analysis techniques. The metrics team should aim at clear, simple and systematic data collection mechanisms. This definition task should be carried on along with the suppliers of data. This can mean the difference between a programme that works and a programme that fails.

3.2.d. Designing storage, analysis and feedback mechanisms

Having determined from where and how to get the raw data needed to be collected for the SMP, the metrics team now considers where to put this data.

With respect to the storage issue, the metrics team considers the metrics database, keeping in mind that the process of designing and implementing this metrics database should be done preferably before the stream of data begins. The composite and base metrics definitions are the main input for designing this database. It is better to keep the first implementation of the database simple, perhaps treating it as a prototype.

With respect to the analysis and feedback issue, the metrics team should consider the type of analysis to perform. Within the programme design, the frequency of the measurement reports and the target audience should be considered. The team members should also take care of the style of the reports. Based on the metrics, they will choose the styles of data representation that suit their different requirements, choosing among the different styles of textual, pictorial or graphical representation . If in doubt, the metrics team can ask an expert statistician. One good style of representation is to mix words with simple graphs. The metrics team should also aim to store and analyze information that can be used to show the effectiveness of the measurement programme itself.

The metrics team should not forget about the software engineers who supplied them with the required information. They will thus give the staff back information they can use, letting them realize that their effort is useful to the organization. One way to do this is through a news sheet to all the teams contributing to the metrics programme. This will leave people well-disposed towards the SMP.

3.3. The Marketing and Business Planning Stream

During this third stream of the Component design stage, key non-technical areas within the SMP are addressed. This stream consists of two main tasks:

3.3.a. Preparing a business plan, and

3.3.b. Preparing a marketing plan.

3.3.a. Preparing a business plan

The issuing of instructions and standards is not enough. People will need help to change and to turn concepts into practice. From this point comes the necessity to support the organization's staff during the implementation of the SMP, in order to have the expected benefits. So, a support function should exist prior to and during metrics implementation. The metrics team members would offer their support to the development teams, that is the software engineers and all other parties involved. So, the metrics team also acts as a metrics support group. It is argued that it will take about 30 days of external support per development team to get an SMP up.

If the organization involved agrees that a support team will exist for implementation, then the metrics team builds this support strategy into the design by treating each development team as an account or client. Within each development team, a facilitator is appointed. He or she manages the implementation of metrics within his or her team.

A contract or service level agreement is drawn up between the metrics support group and the development teams, in which what will be provided by each party and when is clearly stated. Such a document will include specific responsibilities allocated to individuals, for every period of the life of the SMP. As well as documenting what will be done when, this document is also a useful way of focusing the staff's involvement in the measurement programme. The members of the organization's functional groups see what they are getting into.

Although it is good to build this concept of support into the SMP, the metrics team should nonetheless ensure that such a support can be supplied. Therefore, the metrics team will have to look at the available resources and at the number of staff groups within the organization that need to be supported. Other functional groups not directly involved in development may also require support. So, all customer groupings should be accounted for. The business plan will form the corner-stone of the presentation to senior management when seeking for approval to proceed with the SMP.

3.3.b. Preparing a marketing plan

Publicity becomes vital as we move towards the implementation of the measurement programme. Two issues need to be considered, first, launching the implementation phase of the programme, second considering the ongoing publicity for the programme within the organization. Interest and awareness need to be turned into commitment and involvement.

So, there should be a clear planning as to how to sell the metrics initiative to the different customer groups involved. This plan, together with plans covering the ongoing publicity of the SMP in the organization and the feedback to the participants, form the marketing plan.

3.4. The Infrastructure Design Stream

An initiative that is intended to change the way people think, act and work must address the infrastructure that will enable change. This issue is addressed in this fourth stream of the Component design stage, which consists of two main tasks:

3.4.a. Defining the support infrastructure, and

3.4.b. Consolidating and moving the design forward.

3.4.a. Defining the support infrastructure

After the need to support the process of change incurred through the measurement initiative is accepted as a fact, the metrics team has to consider how this support should be provided. If such a support is not provided, people will revert to their previous way of thinking and of working. There are a number of options that answer the "how" of the support function.

An obvious choice is to retain the team that has developed the SMP as a centralized group responsible for the support. Such a group would assume the responsibility for implementing the programme. Furthermore a support group from within the development teams is established. So, for each functional grouping, a local metrics coordinator is appointed. He or she takes on the responsibility of implementing the metrics initiative for his or her group under the guidance and direction of the centralized support group. The intention is to transfer expertise from the central group to these local coordinators as quickly as possible. The ownership of the SMP by the development function within the organization is vital to its success. The advantage of this method is that the organization has and continues to develop a center of expertise in measurement issues, as a single point of reference responsible for the implementation of the measurement initiative and handling the necessary coordination. The disadvantage is that if such a group is external to the software development process, no sense of ownership of the measurement initiative is established.

A second option is to use a centralized group to develop expertise quickly within a small number of development teams, and then to let these new experts spread the gospel to the others. In whatever way it is formed, the support group should be impartial.

On another hand, the metrics support group should be confident about its metrics knowledge. Any material or past practice that can help build this measurement knowledge should be used. Also, training should be given by the support function to the individuals that make up the support infrastructure for the measurement initiative, namely to the local coordinators. The training material will cover any information needed for the SMP's implementation within the organization.

3.4.b. Consolidating and moving the design forward

An integrated design should be taken to management for approval to proceed into the implementation of the measurement initiative. Before this can happen, the design of the SMP should be consolidated.

The design taken forward for review should include, against each requirement, information about the core solution, in-house documentation, the provision of training and on-going support. Elements related to the overall measurement initiative are also vital. The set of proposals relating to the overall SMP include the marketing and business plans covering internal publicity, together with the required education, the implementation strategy, and a proposal for an infrastructure that will support the implementation strategy.

The proposals should be prepared as some form of document. Solutions to requirements should be linked to business needs associated with specific customer groupings within the organization. One approach that seems to be acceptable is to structure the design documentation as a report in the following way:

- An introduction,
- Requirements identification,
- Solutions identification and
- Implementation strategy.

An additional issue to be addressed within this fourth stream of the Component design stage is the cost benefit analysis. This is important when seeking senior management approval. Since there is very little material available from external sources that describes the cost and benefits of a wide-range metrics initiative, this task can be hard to perform. Rather than asking “Can we afford to do this?”, parties involved should alternatively ask “Can we afford not to do this?” as to the use of software metrics. In many cases, the cost benefit analysis can be replaced by ensuring that the metrics team has the commitment of the potential customers of metrics within the organization. This is achieved by directing solutions towards their own specific requirements and by making sure that they are aware of this.

Another additional issue that needs to be addressed within this fourth stream of the Component design stage is the budgeting issue. This task also is hard to perform, and assumptions have to be used as accurately as possible. It is advisable to build a contingency into budgets to cover additional work that may be identified as the programme progresses. The parties involved should keep in mind that the SMP will probably require a relatively high investment during the first period, and that returns are not to be expected before then.

The design should then be presented to senior management, seeking approval to move onwards to the build and implementation stages. The metrics team would use the design report prepared to make a presentation to the approvers. After the design solutions have gone through the approval process, we move to the next stage, the Component build stage.

So, the design stage of the SMP can be seen as the core of the SMP's development. The metrics team should not go into the designing before defining clear requirements. Even with clear requirements, the metrics team should not concentrate upon the technical aspects of the programme, such as the metrics, at the expense of the equally important streams that deal with the marketing of the programme and with its support structure.

4. The Component Build Stage

The metrics team members specified the requirements of the SMP and designed solutions to meet these requirements. The Component build stage is the time to change that design into something solid and working, by building whatever was designed or defined in the previous stages. The members of the metrics team have to set things up so that everything is ready for the implementation stage, basically laying the foundation for implementation, by physically building the components of the SMP. This stage encompasses several activities:

- 4.1. Select the implementation leader(s) and group coordinators,
- 4.2. Document techniques and build training material,
- 4.3. Build the metrics database,
- 4.4. Build data collection mechanisms, and
- 4.5. Review built components.

4.1. Select the implementation leader(s) and group coordinators

This is about the formal selection of an implementation leader (or leaders), and having his or her commitment to be responsible for conducting the implementation stage of the SMP. Local coordinators will also help from within the development teams during the SMP's implementation. It is important to choose the right people, the "doers", not the "talkers", and preferably, senior "doers".

To overcome a possible resistance from the staff, the metrics team has to account for everybody and to give solutions to everybody concerned. The involvement of the staff due to perceived benefits is vital and irreplaceable. The centralized support group, the involved commitment of the implementation leader(s) and of the group coordinators form the core of the SMP. Therefore some time has to be spent to get the right people. If this is done wrong, then future problems will be accumulating.

On another hand, as designed in the marketing plan of the SMP, the metrics team should have maintained an awareness of the measurement initiative within the organization. Now, it should ramp things up as the SMP moves towards the implementation stage. Some form of launch for the SMP is needed, and this should be a visible event. It could be a statement of intent, or a seminar. Whatever the launch vehicle is, the metrics team should aim for visibility within the programme. A good method of achieving this visibility is a management workshop. If the statement of intent is enough or all that the organization is ready for, then the metrics team should go for that. It should be made very clear where the programme is heading to, and what it will require from the staff of the organization before starting the implementation.

4.2. Document techniques and build training material

As designed in the business plan of the SMP, the metrics team should document the design solutions for each collected requirement in the form of a set of guidelines or a code of practice. Such a document is a technical paper that should be concise, readable, coherent, and usable. A standard style for all the SMP's documents could be adopted now. So, the metrics team should spend time planning general layout of standards, along with the sponsor and the MCG. Blending different styles in a single document is a useful way of tackling problems.

The metrics team also needs to provide training to the staff members involved with measurement in the organization. The training will be in both the new concepts and techniques related to the measurement requirements of the SMP, and will rely on the previously produced technical documents. Planning is what makes a course work well. So, the metrics team should plan the timings, the context, the exercises, but always budget accordingly. Such a training should be given by people who know their subject well. So, it is preferable if metrics training is presented by members of the metrics team. Once measurement-based techniques become established within the organization, the responsibility can be passed to the relevant support function. By this time, there will be more people available who have experience in using such techniques and who can give effective training courses.

4.3. Build the metrics database

The metrics team needs to build the previously designed metrics database, since it needs to store collected data, and to report on this data to enable analyzing and taking the appropriate corrective actions at the right time. Such a database has to be developed internally since no metrics database package is available in the market, and if available, it would not be perfectly suited to the needs of a specific organization. The metrics team is going to gain experience and knowledge from the implementation of the measurement initiative that will impact upon the structure of the programme. The database to build should therefore be flexible to accommodate such changes. The development of the metrics database should be treated as a project in its own right. So, the metrics team should aim at metrics database of reasonable quality, that has a flexible design that can be changed as experience and knowledge are gained from the implementation of the SMP.

4.4. Build data collection mechanisms

The data collection mechanisms previously defined should now be built. Data collection represents an overhead to software engineers, and building such mechanisms will take some time. There is bound to be a reluctance on the part of project managers and others to sanction the imposition of yet more such work on their staff. The metrics team will try to overcome this by always associating data collection with benefits. The guiding principle behind data collection is that it should be non-intrusive. Collection systems should also be simple. In the case of already built collection mechanisms, and because the data may have been seldom used for any obvious purpose, the information collected could lack accuracy. Therefore, the metrics team has to re-educate people about the need for accuracy.

4.5. Review built components

Within this stage of the SMP's development, there are numerous items to review. These are the documented techniques, the standards, the training material, the metrics database and the data collection mechanisms. The metrics team should not neglect these reviews, as they cover items that form the public face of the SMP. MCG members can usefully participate in the review process. Walkthroughs or Fagan inspections can be performed. In any case, the act of reviewing is more important than the method used.

The components of the SMP, the database, the feedback mechanisms, the infrastructure, the teams and the training have all been accounted for. In most situations, there is only one thing left to do. Management must now be committed to the implementation of the measurement programme. If the potential users and the managers have been consulted all through and have been kept up to date as to the SMP's development, then this vital step is fairly easy.

How to gain management commitment for implementation depends on the overall setting. A presentation supported by endorsements from the MCG and a detailed report could be one way. In the end, the commitment to implementing an SMP depends on two things. The metrics team has to present a proposal to the business that is coherent, complete and that offers opportunities to realize real benefits. Also, the business has to be ready for it. Senior persons in the organization must be mature enough to realize that there are serious problems in trying to develop software without management control, without a sensible degree of discipline and without clear goals. Finally, the managers must realize that there is nothing for free, nothing 100 percent certain, and unfortunately no off-the-shelf solutions.

If the organization follows the common sense approach of gathering clear requirements, and if it realizes that the metrics and the techniques are only a part of the issue, then it is likely to see real benefits. Once everything is ready and management commitment is gained, then the implementation of the SMP can start.

So, a considerable amount of effort is required to turn clear designs into solutions that are ready to implement. This effort is necessary and the expenditure can not be avoided. As always, it is important to devote effort during the build stage to every component of the measurement programme, and that includes the non-technical aspects, such as training material and documentation.

5. The Implementation Stage

Implementing the SMP is a people oriented issue. In fact, the implementation stage is critical because it involves other people. This is a key point to remember because an SMP is more to do with people than with the metrics and the techniques normally associated with such initiatives. To successfully implement a measurement programme, the metrics team should apply thorough interpersonal skills and should handle the different personalities and the different attitudes of the staff members appropriately.

Tools are a plus, but an SMP can work without tools. Tools do not have to be a prerequisite to improvement. So, a "tools phobia" attitude is not justified. Besides, the tools can only be introduced when a process is well understood, and if the environment is amenable to the application of tools. On another hand, the issue of the resources involved with the SMP should be addressed. Actually, some effort has to be put by the development teams of the organization that is initiating the SMP. The problem is that the amount of effort needed is difficult to quantify. Here, honesty is the best policy to adopt.

The MCG has acted so far as the steering committee for the development of the measurement initiative, a sounding board for possible solutions. Effectively, they have been the recipients of information and they have provided feedback on that information. During the implementation stage, the MCG should become much more pro-active. It now needs to own the programme, it should become the focal point for all metrics issues in the organization. So, the focus moves from the metrics team to the MCG.

The implementation of the SMP will not succeed without the cooperation of the staff. Communication, as well as other aspects of implementation require some effort. The key point is that implementation is not free. In fact, the implementation stage is the most deceptive and risky stage of the whole life cycle of the SMP. The planning, the designing and the construction of the metrics programme have been done during earlier stages and this is simply a question of transferring measurement knowledge from a few areas to every area. The metrics team should sell the concept, train, support and implement all the built components of the SMP as well as maintain contact with the staff teams for a certain period. All this takes time.

Although the implementation of the measurement initiative is costly, it should be fairly easy provided the metrics team has laid the foundation properly during the earlier stages. Even with good preparation, problems still have to be overcome, the most difficult ones relating to people rather than to the technical aspects of the programme.

The whole point of the implementation stage is to turn the customers of the measurement initiative, that is the software engineers of the organization into active participants in the SMP. If the metrics team succeeds in this, the participants will become the owners of the programme, and the metrics team will delegate the operation and continued enhancement of the SMP to them. Members of the metrics team should know when to let go, and this is when the staff members are doing more of the positive talking and the innovations than themselves. At that point, success is achieved.

The implementation of the SMP is mainly about maintaining the SMP, keeping up with the issue, and making sure not to abandon the work before getting to its end. The results of the SMP will not show rapidly, and this is what makes persistence even more crucial.

In case some more measurement requirements remain unaddressed by the first implementation of the SMP, the circle would be closed by going right back to the very beginning, taking up some of these unaddressed measurement requirements, and starting a second phase of the SMP. In other words, do it all again.

Chapter 5

Case Study: The Implementation of a Software Metrics Programme in a Lebanese Company

Recall that the Software Metrics Programme is composed of the following five stages, each of which will be applied to a practical case study in this chapter.

1. The Initiation Stage,
2. The Requirements Definition Stage,
3. The Component Design Stage,
4. The Component Build Stage, and
5. The Implementation Stage.

1. The Initiation Stage

The "customer" organization interested in initiating an SMP in its environment is a Lebanese software development company, of medium size, consisting of a management department composed of 2 persons, a sales and marketing department composed of 3 persons, and a development and technical support team composed of 7 persons. The measurement initiative consultant role is played by myself.

This first stage of the SMP consists of three main activities:

- 1.1. Having an initial management decision,

- 1.2. Assigning management responsibility, and
- 1.3. Performing a feasibility study and presenting the feasibility report to management.

1.1. The initial management decision

The triggers identified in our case are several:

a- productivity of staff concerns,

b- quality of products concerns,

c- concerns about customer complaints, especially delays,

d- concerns about the correctness and completeness of software designs,

e- size, time and cost estimation concerns, and

f- lack of documentation, which makes maintenance hard to perform.

The managers were the initiators of all these points, but the developers also initiated points (b), (d) and (f). The responsibility is delegated to the measurement consultant.

1.2. Assigning management responsibility

The sponsor of the measurement initiative within the company should be:

- someone responsible who has software engineering and software metrics knowledge,

- someone who is aware of the existing problems in the organization and who believes that software measurement can be the solution,

- someone who has enough time to assume such a role seriously and fully,

- someone with authority within the organization and with access to senior management, or at least with a certain degree of seniority and involvement.

We asked our software manager to assume such a role, which he agreed to.

1.3. Performing a feasibility study and presenting the feasibility report to management

In order to answer the question "Is the measurement initiative feasible within our company?", we had to further answer the following questions:

♦ <u>Question</u>

-What happens if we do not adopt any measurement initiative?

<u>Answer</u>

We will go on having projects behind schedule, having employees whose productivity is not assessed, having a hard time maintaining products and having bad estimates, along with some problems with some of our customers.

Actually this answer implies that everything will be done to try to make the initiation and implementation of a measurement initiative feasible within our company. This will of course have an impact on later answers.

♦ <u>Question</u>

-Do we have the necessary time and resources?

<u>Answer</u>

We will appoint some of the staff to this task, or even expand the staff for this purpose. This will keep employees from their routine tasks, but both developers and managers agreed that if performed well, the standards and methodology that will result from the SMP will pay back this cost in future projects.

♦ Question

-Does our staff appreciate the notion of software measurement?

Answer

Some do, and some do not, depending on whether they have been previously exposed to software engineering concepts or not. Those who do not appreciate this notion will need additional effort to get to the appreciation and involvement levels required.

♦ Question

-Do we have the required budget?

Answer

We will have to account for what the measurement initiative will cost us while performing measurement activities. We will also have to account for the additional effort that will be put by our staff in all tasks related to the measurement initiative. The managers were aware of these issues and agreed upon them.

♦ Question

-Do we have an involved sponsor capable of maintaining the measurement initiative?

Answer

Yes. Furthermore, the choice of our sponsor implied that the SMP is going to be supported by higher management.

♦ Question

-What are the benefits of the SMP's initiation?

<u>*Answer*</u>

In a few words, the benefits are to control the software process, the software products and our staff, i.e. to get out of the problems that initially triggered the initiation of the measurement programme.

♦ <u>Question</u>

-Are the benefits really worth the trouble?

<u>*Answer*</u>

We thought they definitely were.

To answer these questions, the measurement sponsor appointed two persons who reported to him during this feasibility study. These two persons had the chance to interact, exchange ideas and hold discussions. The team can not be larger than that because the company as a whole is not a large one. As to the selection criteria for the feasibility team, we chose two persons from among the staff who know the business well and who can get valid information from others effectively. Good systems analysts were thus good candidates. These persons were senior enough to have a feel of the work and of the system in the company. They also had a software engineering and a software metrics knowledge.

The feasibility study team, with the consultant's help, needed to perform an initial subject familiarization with the metrics area and an initial market research in order to accomplish their task.

To perform the initial subject familiarization with the metrics area, the feasibility team had to dig into the available literature, primarily because software measurement was not practiced in the market. They mainly looked in LAU's Stoltzfus library. They had to show that the way in which others have successfully used metrics is applicable to our own organization. So, they had to be aware of the work of others, to map others' work to our own needs within the organization and to present this mapping. The "Best Practices" chapters found in software metrics books were the main sources used while performing these tasks.

As to the initial market research, the feasibility team divided the company's staff into the following functional groups: the managers, marketing and sales staff, development staff and technical support staff. A sample from each of these groups was then selected for an interview, including persons with a negative attitude towards the SMP. They were then interviewed by the consultant and the feasibility team. The consultant is the "metrics expert" and the feasibility team members are the "company experts".

The interview addressed the following areas. First, the interviewers provided a brief introduction to the subject of software metrics, and the reasons why the company is looking at the topic. Essentially, the reasons were the previously identified triggers. This introduction should not be prescriptive in order not to constrain the interviewees, but it provided a reason for the interview and an initial direction.

Second, the problems faced by the interviewee in his or her work were ascertained. One way to get this part of the conversation going was to ask the interviewee briefly to describe his or her role in the organization. This will usually bring up areas that can be explored in more detail, given the high-level requirements placed on the SMP. The managers described their role as providing some sort of control over the flow of work performed in the company. The problems they faced were mainly a lack of information that can enable them to actually assess and control the work performed in the company, from different perspectives (quality-wise, productivity-wise, etc.). Developers described their work as performing any of the activities involved during the software development process. They asked for objective ways to assess and control the quality of the software products they produce, and they wanted to avoid problems with their customers.

Lastly, the interviewees were asked what additional information they feel will be beneficial to the project. The interviewers usually need to drive this part of the session themselves. In brief, the additional information asked for was objective ways of showing the status of projects, products and employees. Most importantly, two points were noted by the feasibility team. Two habits were followed in the company that were particularly relevant to the SMP's initiation. First, the staff used time sheets to plan for future work ahead of time. The time sheets contained a listing of tasks and the anticipated effort to complete each task. For us, the task was the unit of work. The time sheets were assessed weekly. Second, any customer service request, be it a request for fixing a bug or performing an enhancement to any of our software products at a customer's site, installing a new product, fixing data errors, or others, was recorded in a "customer services requests" database by any of the staff who performs such a service. The recording was done through a "customer services requests logging" product.

The results of the interviews were then stored in the SMP repository, a place where we kept and catalogued all information relating to the SMP. In that way, and out of such information, the feasibility report is prepared. The report concluded that "a need for measurement is noted in the organization. The present setting in the company enabled the initiation of the SMP. In fact, people's involvement in the issue is fairly good, and the habits followed were also amenable to measurement". Finally, in order to get authority to proceed, the team got access to management and presented this feasibility report. After this presentation to the managers, the authority to develop the SMP is given.

Note that the initial scope of the SMP is set during this stage, and that this could not have happened unless the initiators, the sponsor, the consultant and the feasibility team listen carefully and constructively to all parties involved.

2. The Requirements Definition Stage

During the previous stage of the SMP, the feasibility team showed that a measurement initiative is feasible in the organization by performing a feasibility study and presenting its outcome to the managers in the form of a report. The authority to develop the SMP is therefore given. The next stage of the SMP is the requirements definition stage, and it consists of several activities.

- 2.1. Initial publicity campaign of the SMP within the organization,
- 2.2. Establish interface through the formation of the Metrics Coordination Group (MCG),
- 2.3. Identify potential super champions,
- 2.4. Obtain specific requirements that will drive the SMP,
- 2.5. Establish initial definitions,
- 2.6. Prioritize the various collected requirements, and
- 2.7. Consolidate the requirements to produce a consolidated requirements specification.

2.1. Initial publicity campaign of the SMP within the organization

All the staff now knows that an SMP is being developed in the company. Everybody's cooperation will be necessary all through the SMP's life, for requirements capture, feedback etc. To stress this point, some sort of marketing of the SMP was needed. So, the metrics team, formed of the consultant, the sponsor and the feasibility team organized informal discussions among all the staff and the managers with a broad topic to discuss, the SMP being developed in the company. Also, they decided to use an in-house newsletter to boost this publicity campaign and to keep stressing that everybody's involvement and cooperation are crucial. Such a newsletter is a chance to thank the staff for every effort they put in the SMP's development. This made the measurement idea more familiar in the company, and allowed the metrics team to identify motivated persons. Everybody was involved in a positive way. Such a campaign required organizing the sessions to be held, and allocating staff time to the attendance of these sessions. Even this fairly small effort was previously accounted for.

2.2. Establish interface through the formation of the Metrics Coordination Group (MCG)

The formation of the Metrics Coordination Group (MCG) was then targeted. Recall that the interface between the metrics team and the rest of the staff within the company is established through the formation of the MCG, acting as the steering committee of the SMP. The MCG has to include representatives from each functional customer grouping. In our case, the MCG included two representatives from the management department, one representative from the sales and marketing department, two representatives from the development team, and one representative from the technical support team, in addition to the sponsor of the measurement initiative, the metrics team and the measurement consultant. Note that members of the MCG and of the metrics team can be the same in companies of this size. Some of the MCG members were less directly concerned than others, like the representative of the marketing team, yet, everybody's inclusion gives the SMP a sense that the issue is a global one and that the solution involves everyone. The choice of these representatives depends on the company's structure and the way it operates. We chose motivated people that think they can help and that the sponsor, the consultant and the managers find self motivated, organized and qualified enough. The MCG members have several tasks to perform, which we restate below:

- ensuring that the SMP addresses the business needs of the organization by demonstrating that all the SMP's customers have been consulted from the start,
- acting as a final review body for any product from the SMP, and
- assessing in periodical meetings, possibly monthly, what happened and what is to happen next in the SMP, in addition to informing others about all these issues.

2.3. Identify potential super champions

The super champions we sought to identify were staff members more interested than others, more motivated than others, and more receptive than others. They come up with ideas, collaborate and feel involved. Knowledge of software engineering concepts and software metrics is a plus. From within the metrics team and the MCG or from outside, these people will be needed all through the SMP's development and implementation.

2.4. Obtain specific requirements that will drive the SMP

This is the core activity of the requirements specification stage. Two techniques can be used to capture the specific requirements within the SMP: a systems analysis interview-based approach and a workshop-oriented brainstorming approach, and these are both illustrated below.

2.4.a. Systems analysis interview-based approach

The measurement consultant and the sponsor, with the help of the metrics team, conducted a systems analysis approach to capture and specify the requirements of the SMP in our company. In the systems analysis interview-based approach, specific key players within the customer set are identified. Structured interviews are then carried out with these individuals. Since our company is a small one, we decided to conduct such interviews with all the staff, at least from the management and the development departments. The interview could be less thorough when applied to some compared to others, but anyway, any person can provide some helpful information, so if time allows, everybody should be interviewed.

A two-pass approach was used. We first aimed at identifying the particular role of the interviewees within the organization as they see it. Managers described their role as providing some control over the software development process and over the staff, as well as performing cost estimation. Developers described their role as performing software development in its broad sense, including time estimation. We then asked the interviewees what responsibilities, in terms of deliverables, they have. Managers and salespersons had, among other, to prepare and negotiate offers with the customers, while developers were responsible for designs, programs, documentation, demonstrations, etc. Finally, the interviewees had to think about what they perceive to be the weaknesses in the processes they operate. Managers did not have clear objective ways of assessing the productivity of their staff and they were sometimes not confident about the time and cost estimates they produce to their customers during bids and proposals. These estimates often proved to be underestimates that resulted in cost and time overruns. Furthermore, they faced a substantial lag problem every time a staff member leaves and someone else has to take over. Managers were also concerned about customer complaints, especially about delays. On the other hand, the developers did not have clear objective ways that enable them to be confident about their software designs and the quality of their products. They wanted to avoid problems with their managers and with their customers. They thus needed to ensure that the software designs they produce are as complete as possible after the design phase of the software process. To do that, they have to extract customer requirements more clearly and more completely and to be able to objectively say that the software designs obtained are as complete as possible. Both developers and managers wanted to avoid the scenario in which projects take more time than expected and designs are altered and expanded during development rather than being well-defined from the start. The second pass of the interviews were merely the formulation, in measurement requirements terms, of the stated demands of the interviewees.

So, the managers and the developers gave us a valuable feedback. Members from the technical support team and the sales and marketing department did not mention any specific concern or problem that they think software metrics can help them solve directly. The collected requirements are presented at the end of the following section.

Usually, only three to four interviews are scheduled in one day, not more. As to the order of performing these interviews, it can be done in several ways. One way is to start by the staff and go up to management. Another method that we followed is to alternate both management and staff interviews to have a combination of two different natures of requirements and to try to take advantage of such a combination.

2.4.b. Brainstorming

The second way of capturing requirements is brainstorming. The brainstorming approach involves a workshop during which the requirements of the SMP are collected. So, the measurement initiative sponsor, the consultant and the metrics team had to plan and coordinate how the workshop will proceed. This required appointing a meeting leader. The sponsor of the SMP was a good candidate since he is familiar with both the subject of software metrics and the company's structure and internal operational mode. The meeting leader has to coordinate events that happen during the workshop as they occur. All the staff members were invited in an official way to the workshop, which was sponsored by the management department. The metrics team along with the sponsor and the consultant worked together and conducted the workshop under the control of the meeting leader, in our case the sponsor. They explained that the aim of the workshop is to obtain a set of requirements that will drive the SMP being initiated in the company. They then started by defining software metrics and introducing benefits of measurement. The definition of software metrics presented in the introductory chapter was explained in depth. Afterwards, the material previously identified in the initial subject familiarization task of the previous stage was used to show examples of the effective use of measurement and to relate information shown to the identified business needs of the organization.

After this introduction is done, the workshop team identified the particular attendants that they thought can help them most in capturing the specific requirements of the SMP. The previously identified "champions" are also focused upon. In our case, these key players are all the members of the management and the development departments. The structure of the workshop session is therefore to present the aim of the workshop and an introduction to the issue, and then to start an interactive process in the form of 'Questions and Answers' discussions during which the workshop team proposes possible requirements to help the staff identify their requirements for the SMP. We used the grouping of metrics requirements presented below during this interactive process. We presented requirements for project metrics, that is metrics that show the status of projects and that enable spotting potential problems. There was also cost, time and size estimation requirements that aim at improving the estimation process. There was quality assessment requirements that enable assessing the quality of software products. There was requirements for productivity metrics that enable assessing the productivity of the staff. There was requirements for design metrics that enable assessing the software designs. We introduced attendants to each of these categories of requirements in turn and in some depth, elaborating more on issues to which they showed special interest. The team had to steer discussions towards requirements definition in the afternoon, in order not to run out of time. The feedback we got enabled us to specify the requirements of the SMP, which we present at the end of the section.

Whatever approach we choose to use, we should end up with a set of specific requirements that have come from the business and that the business wishes the SMP to address. Our identified requirements that will drive the SMP's initiation in our company are presented below.

♦ *The SMP's requirements*

- Requirement 1

Monitor the productivity of our staff in order to assess and improve it,

- Requirement 2

Evaluate the quality of the products being developed in order to assess and control it,

- Requirement 3

Monitor our designs in order to assess and improve them,

- Requirement 4

Evaluate our size, time and cost estimations in order to improve them, and

- Requirement 5

We need to cater for documentation to facilitate later maintenance.

2.5. Establish initial definitions

We defined explicitly what we meant by a design error, an implementation error, an integration error, and a clearly stated requirement or specification versus a requirement or a specification that allows for multiple interpretations. Furthermore, we differentiated between an adjustment and a fault. An adjustment occurs when a modification has to be performed. In this case a minor change will be done to the product. A fault in the product is a misinterpretation of the client's needs, or an incorrect implementation of the client's needs or of the design. In this case a major change has to occur and the responsibility probably lies on the developer. We differentiated as well between a flexible design that allows for later expansion and enhancement and an inflexible design that can not be expanded "naturally".

The necessary agreement to these definitions is obtained by presenting them to the MCG.

As to the software development process model used in our company, it is the waterfall model, in which the following stages are carried out consecutively: requirements stage, specification stage, planning stage, design stage, implementation stage, integration stage and maintenance stage.

2.6. Prioritize the various collected requirements

The prioritization of the collected requirements of the SMP was done essentially by how crucial these requirements are to the company and how easy they will be to satisfy. Two tasks had to be performed to assess the ease of satisfaction of the collected requirements. First, we had to identify the available data sources and to relate the available information to the collected requirements, and second we had to identify the storage, analysis and feedback requirements.

Let us examine the first task, identifying the available data sources and relating this available information to the collected requirements. In our case, among the data available were the previously introduced time sheets and the "customer services requests" database. Recall that the time sheets are put down by the staff members to anticipate their work. Also recall that we used the "customer services requests" database in-house to log all service request reports from all our customers, including maintenance requests, correction requests, enhancement requests, installation requests, etc. Finally a history file containing information about past projects as well as current projects was maintained. At this point, we tried to relate this information to the various collected requirements of the SMP. Time sheets were linked to the productivity requirement (Requirement 1), the customer services requests reports were linked to the productivity requirement and to the quality requirement (Requirement 1 and Requirement 2), and the history file was linked to the estimation requirement (Requirement 4). We were able to get this information by talking to all the parties involved in the company, and we made sure that the data we were using to help us in the prioritization process was actually valid data. For example, we inquired whether the time sheets were not put down by an employee in a way that he or she gives himself or herself unreasonable time for a task to cover up for wasted time. We noted that no available information was kept about our designs and design errors. However, we were all aware of the fact that correcting a design error is among the most expensive tasks to do, and that such errors should be discovered the earliest possible in the process. So we kept our designs assessment requirement (Requirement 3) among the high priority requirements to achieve.

The second task to perform in order to do the prioritization of the collected requirements is to identify storage, analysis and feedback requirements since all requirements associated with an SMP imply the need to store, analyze, and feedback information. We needed to store time sheets, service requests reports, projects history information, and information about our designs. A database was already built for storing information about service requests reports, and we used Microsoft Project to record our time sheets. Handling the storage of the rest of the information was not difficult. We will define pre-formatted sheets that we would fill in to record both project history information and information about our designs. These sheets will accommodate for recording as much useful and relevant information as possible. The analysis of this data was also seen not to be very complicated. We answered questions like "How much data will we have to collect and from how many sources?", "How many customers need to have information fed back, in what form and at what cost?". As a first estimate and mainly because of the medium size of our company, we anticipated that the data to collect will be reasonable in size, the sources of the data will also be manageable in size, and the number of persons who will need feedback is limited. We felt that all our measurement requirements are important, and that their satisfaction is not unfeasible in our environment, so we gave all our requirements a high priority.

2.7. Consolidate the requirements to produce a consolidated requirements specification

Our requirements seemed to be consolidated from their initial capture. However after examining requirement 5, which is the need for documentation to facilitate later maintenance, we noticed that this is not a measurement requirement, but a project management issue. In fact there is no metric that can impose documenting on the staff. Therefore, a management recommendation was communicated, requiring everyone to produce technical documentation of their work, especially prior to leaving the company.

So our consolidated requirements or goals were:

1 ♦ Monitor the productivity of our staff in order to assess and improve it,

2 ♦ Evaluate the quality of the products being developed in order to assess and control it,

3 ♦ Monitor our designs in order to assess and improve them, and

4 ♦ Evaluate our size, time and cost estimations in order to improve them.

We also linked each consolidated requirement to customers. The first and fourth goal were linked to the managers, the second and third were linked to both the managers and the developers.

Finally, these consolidated requirements were reviewed by the measurement sponsor, the consultant and the MCG, so that the company validates the work to be done.

3. The Component Design Stage

During the Requirements definition stage of the SMP, the measurement requirements that will drive the measurement initiative in the company are collected. The design of a solution is now addressed. The third stage of the SMP is the component design stage and it consists of four main streams or sets of activities. These four streams will eventually come together and result in a design proposal for the measurement initiative. The streams are presented below.

- 3.1. The Metrics Definition Stream,
- 3.2. The Administration Design Stream,
- 3.3. The Marketing and Business Planning Stream, and
- 3.4. The Infrastructure Design Stream.

3.1. The Metrics Definition Stream

This stream is the core of the SMP. In fact, at this point, we define our metrics and metrics-based techniques. The metrics definition stream consists of two activities:

3.1.a. Model definition through the Goal-Question-Metric paradigm, and

3.1.b. Identification of external products.

3.1.a. Model definition: Goal-Question-Metric paradigm

At this point, we have in terms of Basili and Rombach's Goal-Question- Metric paradigm (GQM paradigm) the goals defined, and the means by which the questions can be resolved (refer to Chapter 2). The goals are essentially the measurement requirements collected, which are restated below:

1♦ Monitor the productivity of our staff in order to assess and improve it, 2♦ Evaluate the quality of the products being developed in order to assess and control it,

3♦ Monitor our designs in order to assess and improve them, and

4♦Evaluate our size, time and cost estimations in order to improve them.

The customer groups involved with these requirements were the management and the development staff. Our sponsor was to represent his group during the rest of the SMP's life. As to the representatives of the development team, they were also the same persons being members of the MCG and of the metrics team. We felt that these persons while developing and implementing the SMP will show added concern to really express the needs of their own group, which are also their own needs, and to represent the interests of their colleagues as to measurement issues.

Now, we had to start with the question or modeling task of metrics definition. We examined each measurement requirement in turn.

• ***The first requirement*** of the SMP is to "Monitor the productivity of our staff in order to assess and improve it". It is a process-oriented requirement since it relates to the productivity attribute of the process, formed of the productivity of the individual staff members.

Then for this attribute within requirement one, we addressed the following issues. First, we defined productivity as the degree to which economic value is produced. Second, we agreed that we were monitoring productivity, versus predicting it. Lastly, we had to develop a model that will satisfy the requirement. To do that, we looked into the set of software metrics presented in Chapter 3 and found out that productivity is normally expressed as:

Productivity = work product / product cost.

Generally speaking, productivity within software development is expressed as:

Productivity = project size / project effort.

This is the basic productivity model that was used to derive the composite and base metrics. We held weekly meetings to assess the week's work, through monitoring the time sheets previously prepared. So, we chose engineering weeks for project effort. For project size, it was obviously insignificant to use Lines of Code since we developed in a high level relational database language where the LOC concept does not hold as in previous generation languages. Function points do not account for activities that are not implementation activities, like design, analysis, documentation, etc. So, we devised our own measure. The result is given below.

Metric for Requirement 1

"Monitor the productivity of our staff in order to assess and improve it".

- *Productivity* = Conformance to approved time sheets, assessed weekly.

Time sheets were essentially a listing of tasks with the time it takes to perform these tasks, and had to be approved by the project manager concerned. A task is the unit of work we adopted, and can consist of developing a form, a report, a graphical display, conducting testing, producing documentation, performing an analysis function or a design function etc. Therefore, we thought that it was reasonably significant to use such time sheets for measurement purposes. Conformance to time sheets, assessed weekly is a means of monitoring staff's productivity. Employees often behind schedule obviously have a problem, either in predicting the time it takes them to perform a task, or in performing the task. Such persons would need added attention because they may cause a whole project to be behind schedule, with all the consequences that follow.

- ***The second requirement*** of the SMP is to "Evaluate the quality of the products being developed in order to assess and control it". It is a product-oriented requirement since it relates to quality attributes of our software products.

Then for this attribute within requirement two, we addressed the following issues. First, we defined quality as the satisfaction of user or customer requirements. Second, we agreed that we were also monitoring quality. Lastly, we had to develop a model that will satisfy the requirement. The quality attributes we were interested in were reliability and usability. We selected the following metrics from the set of software metrics presented in Chapter 3.

Metrics for Requirement 2

"Evaluate the quality of the products being developed in order to assess and control it".

Reliability *Is the ability of a software product not to fail.*

- ***System test faults*** =

Total number of software faults reported by the testing function during system test / Size of the product.

- ***Customer change requests*** =

Number of unique change requests made by customers during a certain period of field use of a software product release / Size of that release.

- ***Defect density measure*** =

Defects reported from the customer field / Size of the product.

- ***MTTF***, Mean Time To Failure.

MTTF = t_{tot} / n_t where

t_{tot} = total time period of use, and

n_t = number of failures in t_{tot}.

Usability *Is the ease of use of the software product.*

- Number of user complaints, defect, fault, or error reports from the customer field, different than those relating to reliability, per module.

- Readability of the software product's documentation is used to assess how such a documentation assists in the usability of the software product.

Note that in the definition of all these metrics, what we mean by the size of the product is the number of items (forms, reports ...) in the menu forming a specific release of the product. This assumes a fair amount of complexity of all these items.

• ***The third requirement*** is to "Monitor our designs in order to assess and improve them". It is a product-oriented requirement since it relates to the completeness and correctness of our software designs.

Then for this attribute within requirement three, we addressed the following issues. First, we defined a complete, correct design to be a design that accommodates for all the clients' requirements in the most efficient way and that undertakes no or minor changes during the implementation phase. Second, we agreed that we were also monitoring these attributes of the designs. Lastly, we had to develop a model that will satisfy the requirement. Since our requirement was not to assess the complexity of designs, Applied Design Metrics were of no interest to us. What we actually wanted is to be able to know early that our designs are being incomplete, and to be able to remedy to this the earliest possible. Looking at the set of software metrics proposed in Chapter 3, we found that design faults is the metric that suits our requirement.

Metric for Requirement 3

"Monitor our designs in order to assess and improve them".

- ***Design Faults*** = The count of faults found during all design review of a certain software design.

Of course, serious practice of design reviews has to be performed in the company. Actually, we were used to perform such reviews and our designs were fairly good except in one project, and this is what triggered this requirement.

• ***The fourth requirement*** of the SMP is to "Evaluate our size, time and cost estimations in order to improve them". It is a product-oriented requirement since it relates to attributes of our products, namely the goodness and accuracy of estimates.

Then for this attribute within requirement four, we addressed the following issues. First, we defined a good estimate as an estimate that proves to be within a tolerable margin of the real figures. As a start, we set our margin to be between 25% and 50%. Second, we agreed that we were trying to predict these attributes. Lastly, we had to develop a model that will satisfy the requirement. Algorithmic cost estimation models, or model-based strategies for estimation rely on either LOC or FP, which are both not directly applicable to us (also because of the high-level language we used). So, we looked into the set of software metrics presented in Chapter 3, and we found that the following technique-based strategies suit our requirement.

<u>Estimation techniques for Requirement 4</u>

"Evaluate our size, time and cost estimations in order to improve them".

- Expert judgment by analogy using the Delphi technique for reconciliation,
- Bottom-up approach (Functional decomposition), and
- The process of ranking functional entities.

Finally note that in deriving our metrics, we relied heavily on the "state of the art", that is on already available metrics related to our requirements. The process of finding out defined metrics which can be used to suit specific measurement requirements is an important alternative to internal modeling.

<u>3.1.b. Identification of external products</u>

To the four requirements collected, we found no proprietary products available on the open market, or techniques in the public domain that we can "import". In all cases, we felt that the satisfaction of our requirements did not need tools as much as it needed the adoption of a "corrective process" within our company.

3.2. The Administration Design Stream

The first stream of this stage resulted in a set of software metrics to use for the implementation of the SMP in our company. We then considered areas of collection, storage, analysis and feedback of the required data in this second stream of the Component design stage. These are areas needed for a practical use of the defined metrics. This stream consists of four main tasks:

3.2.a. Mapping base metrics to available data,

3.2.b. Establishing links to data administrators,

3.2.c. Defining data collection mechanisms, and

3.2.d. Designing storage, analysis and feedback mechanisms.

3.2.a. Mapping base metrics to available data

The data we needed to collect is presented below.

For the first measurement requirement, we needed to keep time sheets of employees.

For the second measurement requirement, we needed to handle:

- The total number of software faults reported by the testing function during system test,
- The number of unique change requests made by customers during a certain period of field use of a given software product release,
- The defects reported from the customer field, and
- The number of user complaints, defect, fault, or error reports from the customer field per module.

For the third measurement requirement, we needed to know the number of design faults found during software design reviews.

For the fourth measurement requirement, we needed to keep historical information about past projects.

The use of the information recorded in the customer services requests database, and the use of our time sheets will supply most of this information. So, our base metrics were almost all mapped to available data. The additional needed information that was not already supplied was any information about projects that we could later use in our estimations, information about the software faults reported by the testing function during system test, and information about the design faults found during software design reviews.

3.2.b. Establishing links to data administrators

The required information will be queried from the customer services requests database directly, but we needed the members of the development team and the managers to actually record accurately such information in this database. The information related to time sheets had to be supplied by the staff members also. Information about design errors will be supplied by the team performing the design reviews. The information to use as historical data relevant to the estimation process is provided by the initial estimators and by any of the software engineers involved. To formalize the issue of information handling, a management decision was made asking all the staff to cooperate in this sense. Links were therefore formalized. The metrics team members documented their requirements from the data providers, and the data providers formally agreed to these requirements.

3.2.c. Defining data collection mechanisms

The question of "how" to supply data was then addressed. The staff was acquainted with the customer services requests logging package and also with the time sheet forms. They were therefore somehow familiar with the data collection mechanisms in place. As to how often such information should be forwarded, it was agreed that time sheets will be handled weekly and that customers' requests will be logged in as they occur. Pre-formatted sheets were used to help the staff members concerned record information labeled as "design errors information" during design reviews, and also to help in recording any information labeled as "historical project information" related in any sense to future estimations. Examples of data that can be kept about past projects to help in later estimations are the size of the project, the modules of the project, the resources of the project, the estimated cost and planned duration and the updates of these figures, together with the actuals. The main reasons why the actual figures differed from the estimated figures should also be kept, along with the major problems encountered during the project. Project managers can summarize their feeling about a past project, and state any reasons that made this project different than the others. This basic data can prove to be very beneficial later.

3.2.d. Designing storage, analysis and feedback mechanisms

Having determined from where and how to get the raw data, we considered where to put it. We had the customer services requests database already designed and built. We also decided to keep our time sheets on Microsoft Project. We devised the formats of the previously mentioned sheets (used for recording design errors and past project information). So, we had no storage problem.

We then addressed the issue of the measurement reports. We had to expand our customer services requests logging package to add more reporting features. We planned to develop internally the same way we developed to customers. The list of reports and graphical displays that we will need was decided upon. Among these reports were "Services performed by a specific employee during a certain time period", "Services performed for a specific customer during a certain time period", "Distribution of services among all our products" etc. which show the time spent on a specific product and the frequency of the requests made, with details of the tasks performed to satisfy these requests. Reports and graphical displays will be used to mix two styles in our measurement reporting. Graphical displays will mainly be used to show more clearly trends and comparisons. Other possible reports and graphical displays used for analysis and feedback would report about the number of system tests, the number of customer change requests, the defect density measure, MTTF, the number of complaints per module and the number of design faults, per module, during a certain time interval.

3.3. The Marketing and Business Planning Stream

During this third stream of the Component design stage, key non-technical areas within the SMP are addressed. This stream consists of two main tasks:

3.3.a. Preparing a business plan, and

3.3.b. Preparing a marketing plan.

3.3.a. Preparing a business plan

At this point, we addressed the issue of the support function. The metrics team had to support both our developers and our managers to enable them to get into the spirit of implementing the SMP. Support will have to be provided in recording specific information about the faults that come up during testing functions. For an efficient and complete use of the customer services requests logging product, support will be given as a form of a detailed user manual for this product. How to conduct software design reviews and how to deal with the faults found in such reviews was extensively dealt with. Key information to keep about past projects was emphasized and made public to everyone. Such information could prove very useful during future estimations. Finally, the metrics team looked into the literature to learn more about the technique-based strategies for performing the cost and time estimation tasks involved in our SMP. The metrics team also provided a report of their acquired knowledge in this subject to the rest of the staff. The facilitators within the development team and the management team were once again the same persons involved in the MCG and the metrics team. A contract or service level agreement was drawn up between the metrics support group and the staff, in which what will be provided by each party and when is stated. So a support function that the company can supply was designed in some detail.

3.3.b. Preparing a marketing plan

We handled publicity by holding a meeting in which we stated what we already achieved, the targets we have and the metrics we devised to answer these targets. We also emphasized on the help of others, especially in providing the required data. We asked whether any of the staff had any comment concerning the definition of the SMP, and whether they agree that the metrics we are proposing address their needs. We finally planned the ongoing publicity of the SMP and the feedback to the participants. In fact, periodical reports will be issued by the metrics team to everyone in the company to keep them informed of our measurement activities. Similar meetings will also be held later to keep up with the subject.

3.4. The Infrastructure Design Stream

The fourth stream of the Component design stage consists of two main tasks:

3.4.a. Defining the support infrastructure, and

3.4.b. Consolidating and moving the design forward.

3.4.a. Defining the support infrastructure

We now addressed the issue of how the support function should be provided. From among the number of options that answer the "how" of the support function, we decided to retain the team that has developed the SMP as a centralized group responsible for the support function. Such a group would assume the responsibility for implementing the programme and would be a center of expertise in measurement issues in the company. Since our development team was not a large one, we did not have to deal with several functional groups etc. What we aimed at achieving was to transfer expertise from the support group to the rest of the developers. So, local metrics coordinators were appointed from within the development team whose role was also to assume the responsibility of implementing the metrics initiative within the group under the guidance and direction of the support team. We were trying to establish a sense of ownership of the SMP by the developers.

We held a measurement meeting that gathered the consultant, the sponsor and the metrics team. We rose the issue of what we should know ourselves as to the metrics subject in order to be able to support others. Actually we had previously gained substantial such knowledge, mainly from the literature and from devising our own standards. We agreed that we should communicate this knowledge to the metrics coordinators. We wanted measurement knowledge to spread over to everyone.

3.4.b. Consolidating and moving the design forward

We consolidated our design by providing for each of our measurement requirements, information about the core solution, in-house documentation, the provision of training and on-going support. We also included in the consolidated design the marketing and business plans covering internal publicity, the implementation strategy, and a proposal for an infrastructure that will support the implementation strategy.

We then presented this consolidated design to senior management, seeking for approval to move onwards to the build and implementation stages of the SMP. Our report was composed of an introduction, a requirements identification, a solutions identification, and an implementation strategy. Our presentation included a briefing of what triggered the SMP, our measurement requirements, and a complete design made up of the proposed metrics solutions and the way to implement and support these solutions.

As to the cost benefit analysis, we asked "Can we afford not to adopt this measurement solution?", instead of asking "Can we afford to adopt this measurement solution?" We ensured that we had the commitment of all our staff within the company for the SMP's implementation. As to the budgeting issue, we tried to budget as accurately as possible for our SMP, and we built a contingency into our budgets to cover additional work that may be identified as the measurement programme progresses. A comprehensive measurement design solution was in place.

4. The Component Build Stage

During the previous stage, solutions to the SMP's requirements were designed. During this fourth stage, these solutions will be built. The metrics team has to set things up so that everything is ready for the implementation stage by physically building the components of the SMP.

The Component build stage consists of several activities.

- 4.1. Select the implementation leader(s) and group coordinators,
- 4.2. Document techniques and build training material,
- 4.3. Build metrics database,
- 4.4. Build data collection mechanisms, and
- 4.5. Review built components.

4.1. Select the implementation leader(s) and group coordinators

Our implementation leaders were the members of our metrics team. They worked under the supervision of the SMP's sponsor. During the past stages, we got an idea of whom from among the rest of the staff can also help implementing the measurement initiative. These persons could be added to the already appointed local coordinators. The commitment of the implementation leaders and coordinators is very crucial. How serious will the SMP's implementation be depends on this commitment.

An awareness of the measurement initiative within the company has been maintained through the previous development stages. We now needed some form of a visible launch of the SMP's implementation stage. We prepared a report summing up the status of the SMP at our company, covering the designed solutions. During the launch session, attended by all the staff, we presented this report concluding that we were getting ready for the implementation of the SMP. We emphasized on what is going to be needed from all the staff in order to get them prepared to the fact that a change is being prepared for and is about to happen in our company.

4.2. Document techniques and build training material

We then addressed the documentation issue. In order to be complete, a solution within the SMP should be clearly documented. So, for each of the four collected measurement requirements that drove our metrics initiative, we produced a documentation work, covering the associated design solutions, and providing the necessary guidelines. Such a document is a technical paper, and we made sure that it is concise, readable, coherent, and usable. We pursued consistency in style among all our documents.

Our first measurement requirement is to "monitor the productivity of our staff in order to assess and improve it". Its solution requires handling time sheets accurately by reflecting any changes in tasks or any justifiable delays in these time sheets. Such time sheets are assessed every week.

The second measurement requirement is to "evaluate the quality of the products being developed in order to assess and control it". Its solution requires a consistent use of our customer services requests logging product. It also assumes the existence of a system test function during which faults are recorded appropriately. Both these issues were documented in our documentation material.

Our third measurement requirement is to "monitor our designs in order to assess and improve them". Its solution involves conducting design reviews appropriately. Our documentation therefore included guidelines as to how to conduct design reviews.

Finally, our fourth measurement requirement is to "evaluate our size, time and cost estimations in order to improve them". Its solution requires mastering technique-based strategies of cost and time estimation. So, our manuals documented the expert judgment by analogy that uses the Delphi technique for reconciliation, the bottom-up approach that deals with functional decomposition, and the process of ranking functional entities.

This documentation was handed over to all the staff members for them to read carefully. We held global informal sessions a few days after distributing this material to discuss the measurement solutions documented and to clarify any points that are unclear to any of the staff members.

4.3. Build a metrics database

Our metrics database was already built, so was our metrics package, i.e. the Service Requests Logging product. We made sure that the design of this database was flexible enough to accommodate further changes that may occur while we are implementing the SMP.

4.4. Build data collection mechanisms

Data collection mechanisms were accounted for by adding to our customer services requests logging package the previously agreed upon list of new reports and graphical displays. We kept associating these data collection mechanisms with benefits, in order to avoid any reluctance from the staff who may view these activities as keeping them away from their daily tasks, and therefore a waste of time. Mechanisms to collect further data from the design reviews and the technique-based estimation strategies were also handled by printing and making available the previously agreed upon pre-formatted sheets.

4.5. Review built components

The metrics team along with the sponsor and the MCG reviewed the documented techniques, the standards, the training material, the metrics database and application, and the data collection mechanisms. For us, these were the public face of the SMP, and we insisted on a "good" face.

After all the above components are built, there remains one thing to do. The managers must now be truly committed to the implementation of the Software Measurement Programme. To gain management commitment for implementation, we made a presentation supported by endorsements from the MCG and a detailed report. We felt that presenting the business a proposal that is coherent and complete and that offers the business the opportunity to realize real benefits is the only way to get management's approval. This is what we were doing all through and this is what we presented now.

5. The Implementation Stage

Since implementing the SMP is a people oriented issue, we applied our interpersonal skills and tried to handle the different personalities and the different attitudes as much as we could. We knew there was little to achieve without good communication among everybody, so we insisted on developing the best communication channels with each other.

We tried as much as possible to account for the resources needed, especially the time of our metrics team and sponsor. The amount of effort needed to set up the SMP is not easily quantified, so we handled the resources point with extra care.

We kept up with the measurement issue, ensuring that we're providing the necessary support. We tried to make sure that the SMP's implementation is not being forgotten. So, we held weekly meetings to examine the time sheets. Similarly, we held monthly "measurement meetings". We thus issued reports from our metrics database monthly which we made available to management. After examining and analyzing these reports, the managers decide on actions to take in order to get to the improvement goals we had set in several fields. The managers also used measurement data to decide on any corrective actions to be taken. We monitored closely employees and products in which we saw possible future problems. Similarly, we made sure to show some appreciation to those of the staff whose performance was positive or getting better from month to month. Above all, we tried to keep everybody motivated towards that subject.

The point of the implementation stage of the SMP is to turn all the staff into participants in the programme, so that they become the owners of the programme. At that point, we could say that we achieved success by delegating the SMP's operation and continued enhancement to all our staff. A well implemented SMP will pay back its cost in the form of a more controlled software process, with all the benefits that such a control entails. So, after focusing on finding, designing and building the right measurement solutions, the implementation stage is about implementing these solutions well.

The SMP's implementation sets up a closed-loop feedback mechanism within which monitoring and corrective actions will lead to incrementally improving the software development and maintenance process over time. It is all about keeping up the good work in order to reach such results.

Chapter 6

Conclusion

A lot has been written about the importance of software measurement. This importance has nonetheless not been reflected through the implementation of software measurement initiatives in small and medium companies. The issue has always been that, although useful, implementing a software metrics initiative is very tedious and inherently complex. This work aims at showing that the implementation of a measurement initiative is in fact very useful, but that it is also feasible. Initiating and implementing an SMP requires extensive effort, and planning, and everything it takes to successfully manage a process. But it is a systematic process that software practitioners can follow.

Software metrics in isolation are useless. Only when they form part of the enabling function within an organization's will to improve its process and its products can software metrics deliver their full potential. Measurement simply puts you in a position where you have the chance to solve problems. But you will have to solve your problems yourself.

Napoleon said that *"The right information at the right time is nine-tenths of any battle"*.

In the domain of software production, a well defined, well designed, well built and well implemented Software Metrics Programme can provide software practitioners with the right information at the right time, to help them win their own battle. This may lead us a bit closer to finding the "silver bullet that would slay the wolf".

References

Basili, V. R. and Rombach, H. D. 1988. The TAME Project: Towards Improvement-Oriented Software Environments. *IEEE Transactions on Software Engineering,* vol. 14, no. 6, (June), pp. 758-773.

Gilb, T. 1988. *Principles of Software Engineering Management.* Addison-Wesley Publishing Company.

Gillies, A. C. 1992. *Software Quality: Theory and Management.* Chapman and Hall Computing.

Goodman, P. 1993. *Practical Implementation of Software Metrics.* International Software Quality Assurance Series, Q&A Forum, McGraw-Hill Book Company.

McDermid, J. 1991. *Software Engineer's Reference Book.* Butterworth Heinemann.

Moller, K. H. and Paulish, D. J. 1993. *Software Metrics, a Practitioner's Guide to Improved Product Development,* First Edition. IEEE Computer Society Press, Chapman and Hall Computing.

Rae, A. and Robert, P. 1995. *Software Evaluation for Certification, Principles, Practice and Legal Liability*. Haw-Ludwig Haussen Quality Forum. The McGraw Hill International Software Quality Assurance Series.

Schach, S. R. 1993. *Software Engineering*, Second Edition. Asken Associates Incorporated Publishers.

Sommerville, I. 1992. *Software Engineering*, Fourth Edition. Addison-Wesley Publishing Company.

Bibliography

Arthur, L. J. 1993. *Improving Software Quality, An Insider's Guide to TQM.* Wiley Series in Software Engineering Practice, Wiley Professional Computing.

Australian Standard, New Zealand Standard: Quality Management and Quality Assurance Standards, 1993.

Basili, V. R. and Rombach, H. D. 1988. The TAME Project: Towards Improvement-Oriented Software Environments. *IEEE Transactions on Software Engineering,* vol. 14, no. 6, (June), pp. 758-773.

Bollinger, T. 1995. What Can Happen When Metrics Make the Call. *IEEE Software,* (January), p. 15.

Boloix, G. and Robillard, P. N. 1994. *Comprehensive Software Metrics Framework.* Departement de Genie Electrique et Genie Informatique, Serie Informatique, Ecole Polytechnique de Montreal.

Brooks, F. P. 1987. No Silver Bullet, Essence and Accidents of Software Engineering. *Computer,* (April), pp. 10-19.

Burgess, A. 1995. Mad About or Mad at Measurement? *IEEE Software,* (January), pp. 115-116.

Churcher, N. I. and Shepperd, M. J. 1995. Correspondence, Comments on "A Metrics Suite for Object Oriented Design". *IEEE Transactions on Software Engineering,* vol. 21, no. 3, (March), pp. 263-265.

Cook, R. 1995. Real-World Lessons in Software Metrics. *IEEE Software,* (July), pp. 109-110.

Drake, T. 1996. Measuring Software Quality: A Case Study. *Computer,* (November), pp. 78-87.

Fenton, N. E. 1991. *Software Metrics, A Rigorous Approach.* Chapman and Hall.

Gilb, T. 1988. *Principles of Software Engineering Management.* Addison-Wesley Publishing Company.

Gillies, A. C. 1992. *Software Quality: Theory and Management.* Chapman and Hall Computing.

Goodman, P. 1993. *Practical Implementation of Software Metrics.* International Software Quality Assurance Series, Q&A Forum, McGraw-Hill Book Company.

Grady, R. B. 1992. *Practical Software Metrics for Project Management and Process Improvement.* Hewlett-Packard Professional Books, Prentice Hall, Inc.

IEEE Standards Collection, Software Engineering, 1994 Edition.

Ince, D. et al. 1993. *Introduction to Software Project Management and Quality Assurance.* The McGraw-Hill International Series in Software Engineering.

Jones, C. 1991. *Applied Software Measurement, Assuring Productivity and Quality.* Software Engineering Series, Mc-Graw Hill, Inc.

Jones, K. 1993. *Automated Software Quality Measurement.* Van Nostrand Reinhold.

Jorgensen, M. 1995. Experience with the Accuracy of Software Maintenance Task Effort Prediction Models. *IEEE Transactions on Software Engineering,* vol. 21, no. 8, (August), pp. 674-681.

Krishnamoorthy, S. and Fisher, D. 1995. Machine Learning Approaches to Estimating Software Development Effort. *IEEE Transactions on Software Engineering,* vol. 21, no. 2, (February), pp. 126-136.

Lorenz, M. and Kidd, J. 1994. *Object-Oriented Software Metrics.* Prentice Hall Object-Oriented Series.

Macro, A. 1990. *Software Engineering, Concepts and Management.* Practical Software Engineering Series, Prentice Hall, Inc.

McDermid, J. 1991. *Software Engineer's Reference Book.* Butterworth Heinemann.

Moller, K. H. and Paulish, D. J. 1993. *Software Metrics, a Practitioner's Guide to Improved Product Development,* First Edition. IEEE Computer Society Press, Chapman and Hall Computing.

Novak, G. S. Jr. 1995. Conversion of Units of Measurement. *IEEE Transactions on Software Engineering,* vol. 21, no. 8, (August), pp. 651-661.

Paulk, M. C. et al. 1995. *The Capability Maturity Model: Guidelines for Improving the Software Process.* Carneige Mellon University, Software Engineering Institute, The SEI Series in Software Engineering, Addison-Wesley Publishing Company, Inc.

Perlis, A., Sayward, F., and Shaw, M. 1983. *Software Metrics: An Analysis and Evaluation.* The MIT Press.

Putnam, L. H. and Myers, W. 1992. *Measures for Excellence, Reliable Software on Time, Within Budget.* Yourdon Press Computing Series, Prentice Hall, Inc.

Rae, A. and Robert, P. 1995. *Software Evaluation for Certification, Principles, Practice and Legal Liability.* Haw-Ludwig Haussen Quality Forum. The McGraw Hill International Software Quality Assurance Series.

Rook, P. 1990. *Software Reliability Handbook.* Elsevier Applied Science.

Schach, S. R. 1993. *Software Engineering*, Second Edition. Asken Associates Incorporated Publishers.

Shepperd, M. 1995. *Foundations of Software Measurement.* Prentice Hall, Inc.

Shepperd, M. 1993. *Software Engineering Metrics. Volume 1: Measures and Validations.* The McGraw-Hill International Series in Software Engineering.

Sommerville, I. 1992. *Software Engineering*, Fourth Edition. Addison-Wesley Publishing Company.

Tian, J., Peng, L., and Palma, J. 1995. Test-Execution-Based Reliability Measurement and Modeling for Large Commercial Software. *IEEE Transactions on Software Engineering,* vol. 21, no. 5, (May), pp. 405-414.

Tian, J. and Zelkowitz, M. V. 1995. Complexity Measure Evaluation and Selection. *IEEE Transactions on Software Engineering,* vol. 21, no. 8, (August), pp. 641-650.

Weinberg, G. M. 1993. *Quality Software Management, Volume 2, First Order Measurement*. Dorset House Publishing.