Software Modeling and Measurement: The Goal/Question/Metric Paradigm[1]

Victor R. Basili

Institute for Advanced Computer Studies
Department of Computer Science
University of Maryland

Abstract

This paper discusses the use of the Goal/Question/Metric paradigm as a mechanism for defining and interpreting software measurement. Templates are provided for defining goals and generating questions. Different types of metrics are discussed. Examples of both process and product goals are defined.

Introduction

Any engineering process requires feedback and evaluation. Software development is an engineering discipline and measurement is an ideal mechanism for feedback and evaluation. The measurement and information fed back to all parties, e.g., developers, managers, customers and the corporation, helps in the understanding and control of the software processes and products, and the relationships between them. It helps in making intelligent decisions and improving over time. But measurement must be focused, based upon goals and models. We need to establish goals for the various software processes and products and these goals should be measurable, driven by the appropriate models.

There are a large variety of software goals, defined from a variety of perspectives, including the customer, the project, and the corporation. Sample customer goals include customer satisfaction and that the product contains needed functionality. Sample project goals include the need for a high quality process and on time delivery. Sample corporate goals include that the product be salable and that the quality of the software development process improve over time.

There are a variety of reasons for measuring the software development process and product. Measurement is a mechanism for creating a corporate memory and an aid in answering a variety of questions associated with any software development. It helps support project planning, (e.g., how much will a new project cost?); allows us to determine the strengths and weaknesses of the current process and product, (e.g., are certain types of errors commonplace?); provides a rationale for adopting/refining techniques, (e.g., what techniques will minimize current problems?); allows us to assess the impact of techniques, (e.g., does functional testing minimize certain error classes?); evaluate the quality of the process/product, (e.g., what is the reliability of the product after delivery?) and the functionality and user friendliness (e.g., to determine if the system is easy to use and does what the user wants it to do.)

Measurement must be defined in a top down fashion, bottom-up approach won't work.
There are a large variety of software metrics: calendar time, number of open problems, cyclomatic complexity, lines of code/module, number of defects found in inspections, severity of failures, total effort, total number of defects, machine time, lines of code/staff month, total lines of code number of failures during system test. But which metrics does one use and how does one interpret

[1]This work was supported by NASA/GSFC contract NSG-5123 and AFOSR contract 90-0031.

them without the appropriate models and goals surrounding them?

There are a variety of mechanisms for defining measurable goals that have appeared in the literature: the Quality Function Deployment Approach (QFD) [KoAk83], the Goal/Question/Metric Paradigm (GQM) [BaWe84, BaSe84, BaRo88, Ba90b], and the Software Quality Metrics Approach (SQM) [BoBrLi76, McRiWa77]. In this paper we will discuss the GQM approach to measurement. See Appendix for a comparison.

THE GQM PARADIGM

For an organization to measure in a purposeful way requires that it (1) specifies the goals for itself and its projects, (2) traces those goals to the data that are intended to define these goals operationally, and (3) provides a framework for interpreting the data understand the goals. Thus, it is important to make clear, at least in general terms, what informational needs the organization has, so that these needs for information can be quantified whenever possible, and the quantified information can be analyzed as to whether or not the goals are achieved. We use the Goal/Question/Metric (GQM) Paradigm to support a tractable software engineering process.

The Goal/Question/Metric paradigm is a mechanism for defining and evaluating a set of operational goals, using measurement. It represents a systematic approach for tailoring and integrating goals with models of the software processes, products and quality perspectives of interest, based upon the specific needs of the project and the organization.

The goals are defined in an operational, tractable way by refining them into a set of quantifiable questions that are used to extract the appropriate information from the models. The questions and models, in turn, define a specific set of metrics and data for collection and provide a framework for interpretation.

The GQM paradigm was originally developed for evaluating defects for a set of projects in the NASA/GSFC environment. The application involved a set of case study experiments [BaWe84]. It was then expanded to include various types of experimental approaches, including controlled experiments [BaSe84].

Although the GQM was originally used to define and evaluate goals for a particular project in a particular environment, its use has been expanded to a larger context. It is used as the goal setting step in an evolutionary improvement paradigm tailored for a software development organization, the Quality Improvement Paradigm (QIP), and an organizational approach for building software competencies and supplying them to projects, the Experience Factory (EF).

The Quality Improvement Paradigm [Ba85a, Ba85b,BaRo88,Ba89] involves the following steps:
Planning: an iterative process involving characterizing the current project and its environment, setting the quantifiable goals for successful project performance and improvement, and choosing the appropriate process model and supporting methods and tools for this project.

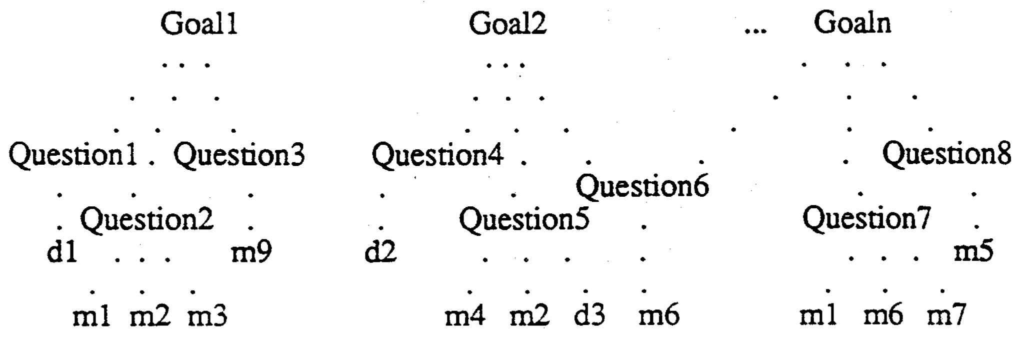
Execution: a closed-loop project cycle which involves executing the processes, constructing the products, collecting and validating the prescribed data, and analyzing it in real-time to provide feedback for corrective action on the current project.

Analysis and Packaging: a post mortem analysis of the data and information gathered to evaluate the current practices, determine problems, record findings, and make recommendations for future project improvements, and a packaging of the experience gained in the form of updated and refined models and other forms of structured knowledge gained from this and prior projects and the storing of the packages in an experience base so it is available for future projects.

The Experience Factory [Ba89] is a logical and/or physical organization that supports project developments by analyzing and synthesizing all kinds of experience, acting as a repository for such experience, and supplying that experience to various projects on demand. It packages experience by building informal, formal or schematized, and productized models and measures of various software processes, products, and other forms of knowledge via people, documents, and automated support.

Thus, we can use the GQM for long range corporate goal setting and evaluation. We can improve our evaluation of a project by analyzing it in the context of several other projects. We can expand our level of feedback and learning by defining the appropriate synthesis procedure for transforming lower-level information into higher-level packages of experience. As part of the QIP we can learn more about the definition and application of the GQM in a formal way, just as we would learn about any other experiences.

The flow from the goals to the metrics in the GQM paradigm can be viewed as a directed graph (the flow is from the goal nodes to the question nodes to the metric nodes):



Here there are n goals shown and each goal generates a set of quantifiable questions that attempt to define and quantify the specific goal which represents an entry node in the directed graph. These questions are based upon a particular set of process, product, and quality models that are not explicitly represented in the graph. Each directed sub-graph reachable from a goal node represents a particular GQM model.

The goal is only as well-defined as the questions it generates and the models on which those questions are based. Since models are often hard to define, they may exist only implicitly in the questions. The more formal, explicit, and complete the models, the more effective the questions and the definition of the goals. Each question generates a set of metrics (mi) or distributions (di). Again, the question can only be answered relative to and as completely as the available metrics and distributions allow. As is shown in the above diagram, the same questions can be used to define multiple goals (e.g. Question6), and metrics and distributions can be used to answer more than one question. Thus questions and metrics are used in several contexts.

The paradigm is used not just for focusing management, engineering and quality assurance interests, but also for interpreting the questions and the metrics. For example, m6 is collected in two contexts and possibly for two different reasons. Question6 may ask for the size of the product (m6) as part of the goal to model productivity (Goal2). But m6 (size of the product) may also be used as part of a question about the complexity of the product (e.g. Question7) related to a goal on ease of modification (e.g. Goaln).

If a measure cannot be taken but is part of the definition of the question, it is important that it be included in the GQM paradigm. This is so that the other metrics that answer the question can be viewed in the proper context and the question interpreted with the appropriate limitations. The same is clearly true for questions being asked that may not be answerable with the data available.

Although there may be many goals and even many questions, the metrics do not grow at the same rate as the goals and questions. Thus a set of metrics could be collected for characterizing the software process and product that will allow us to answer many questions generated by different goals.

THE GQM PROCESS

Applying the GQM involves (1) developing a set of corporate, division and project goals for productivity and quality, e.g., customer satisfaction, on-time delivery, improved quality, (2) generating questions (based upon models) that define those goals as completely as possible in a quantifiable way, (3) specifying the measures needed to be collected to answer those questions and to track process and product conformance to the goals, (4) developing mechanisms for data collection, and (5) collecting, validating and analyzing the data in real time to provide feedback to projects for corrective action and analyzing the data in a post mortem fashion to assess conformance to the goals and make recommendations for future improvements.

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 has been developed [BaRo88]. These templates and guidelines reflect our experience from having applied the GQM paradigm in a variety of environments. The current set of templates and guidelines represent our current thinking and well may change over time as our experience grows.

Goals may be defined for any object, for a variety of reasons, with respect to various models of quality, from various points of view, relative to a particular environment. The goal is defined by filling in a set of values for the various parameters in the template. Template parameters include purpose (what object and why), perspective (what aspect and who) and the environmental characteristics (where).

Purpose:
Analyze some
    (objects: processes, products, other experience models)
for the purpose of
    (why: characterization, evaluation, prediction, motivation, improvement)

Perspective:
with respect to
    (focus: cost, correctness, defect removal, changes, reliability, user friendliness,...)
from the point of view of
    (who: user, customer, manager, developer, corporation,...)

Environment:
in the following context
(problem factors, people factors, resource factors, process factors,...)

Example: Analyze the (system testing method) for the purpose of (evaluation) with respect to a model of (defect removal effectiveness) from the point of view of the (developer) in the following context: the standard NASA/GSFC environment, i.e., process model (SEL version of the waterfall model,...), application (ground support software for satellites), machine (running on a DEC 780 under VMS), etc.

The purpose is meant to define the object or objects of study, what we are going to do and why we are doing it. There may be several objects and we may be doing it for several purposes. It is clear that the author must avoid complex objectives. In some cases it may be wise to break a complex goal into several simpler goals.

The perspective is meant to define a particular angle or set of angles for evaluation. The author may choose more than one model, e.g. defects and changes, and more than one point of view, e.g., the corporation and the project manager. The author should define the model and put himself/herself in the mind set of the person who wants to know the information so that all aspects of the evaluation are performed from that point of view.

The purpose of the environment is to define the context of the study by defining all aspects of the project so it can be categorized correctly and the appropriate set of similar projects found as a basis of comparison. Types of factors include: process factors, people factors, problem factors, methods, tools, constraints, etc. [Ba81a]. In general, the environment should include all those factors that may be common among all projects and become part of the data base for future comparisons. Thus the environmental factors, rather than the values associated with these factors, should be consistent across several goals within the project and the organization. Some factors may have already been specified as part of the particular object or model under study and thus appear there in greater depth and granularity.

There is an order to defining a goal. For example, first one decides upon the object of study. It is assumed that there exists an appropriate model of that object. Then one determines why that object is being studied. For example, if it is to characterize the object, then all that is required is a set of models of the characteristics of interest. If the reason is to evaluate the object with respect to a certain set of qualities, then an evaluative model must be chosen along with an evaluation algorithm. In a sense, the why limits the set of focus models available. The focus model is chosen in the context of the who, i.e. the focus may change if the who is the project manager, requiring immediate feedback, versus, if the who is the corporation and a long range evaluation might be acceptable.

Different sets of guidelines exist for each of the different objects of study, i.e., there are product-related and process-related questions based upon product and process models.

For each product under study there are three major areas that need to be addressed: (1) definition of the product (purpose), (2) definition of the quality perspectives of interest (perspective), and (3) feedback related to the quality perspectives of interest.

(1) Definition of the product defines a model of all those aspects that characterize the particular product under study. It includes questions related to:

logical/physical attributes (a quantitative characterization of the product in terms of the logical attributes such as function, application domain, etc. and physical attributes such as size, complexity, etc.),

cost (a quantitative characterization of the resources expended related to this product in terms of effort, computer time, etc.), *changes and defects* (a quantitative characterization of the errors, faults, failures, adaptations, and enhancements related to this product), and

context (a quantitative characterization of the customer community using this product and their operational profiles).

(2) The quality perspectives of interest are based upon the focuses of interest for the product. The perspective should be based upon some model of the product that provides a framework for measurement. The models used here may be mathematically tractable models or qualitative models. Quality perspectives of interest (e.g., reliability, user friendliness), include questions related to

the *major model(s)* used (a quantitative specification of the quality perspective of interest),

the *validity of the model* for the particular environment (an analysis of the appropriateness of the model for the particular project environment),

the *validity of the data* collected (an analysis of the quality of data), and optionally,

a *substantiation of the model* (an alternative model to help evaluate whether the results of the primary model are reasonable).

This last option is taken when there is some concern about the validity of the primary model or the data.

(3) Feedback includes questions related to improving the product relative to the quality perspective of interest (a quantitative characterization of the product quality, major problems regarding the quality perspective of interest, and suggestions for improvement during the ongoing project as well as during future projects). It should also include things learned with regard to process, application and other products based upon what we have learned here.

Feedback very often requires reference to other factors not explicitly mentioned in the definition of the product or perspective. In these cases it should be checked that these factors exist either in the environment section (when there is an attempt to evaluate against a data base) or in the definition of the product section (when there is a need to examine the model of the project).

For each process under study, there are three major areas that need to be addressed: (1) definition of the process, (2) definition of the quality perspectives of interest, and (3) feedback from using this process relative to the quality perspective of interest.

(1) Definition of the process includes questions related to

process conformance (a quantitative characterization of the process and an assessment of how well it is performed), and

domain conformance (a quantitative characterization of the object to which the process is applied and an analysis of the process performer's knowledge concerning this object and its domain by the process performers).

(2) Quality perspectives of interest follows a pattern similar to the corresponding product-oriented subgoal including, for each quality perspective of interest (e.g., reduction of defects, cost effectiveness), questions related to the major model(s) used, the validity of the model for the particular environment, the validity of the data collected, the model effectiveness and the

substantiation of the model).

(3) Feedback follows a pattern similar to the corresponding product-oriented subgoal.

VIEWS OF METRICS

Metrics can be objective and subjective. An objective metric is an absolute measure taken on the product or process. Examples include: time for development, number of lines of code, work productivity, number of errors or changes. Objective metrics are usually based upon an interval or ratio scale. Subjective metrics represent an estimate of extent or degree in the application of some technique or a classification or qualification of problem or experience. They are used in situations where there is no exact measurement, usually on a relative scale. Examples include: the degree of use of a method or technique or the experience of the programmers in the application. Subjective metrics are usually based upon a nominal or ordinal scale.

Measures may be taken of the product and the process. Product measurement is on a developed product or document, i.e., source code, object code, requirements document. Examples include lines of code and readability of the source code. Process measurement is taken on the activities used in developing the product. Examples include the use of a method or the effort expended in staff months.

We can measure cost and quality. Cost includes the measure of any resource expenditure used in a project, e.g., staff months, computer time, hardware cost, purchased software, calendar time. Quality is a measure of some form of value of the product or process, e.g., reliability, functionality, ease of change, correctness, reusable components developed.

The choice of metrics is determined by the quantifiable questions which are based upon the models used. The guidelines for questions acknowledge the need for generally more than one metric, for both 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 organization to organization. This reflects their being tailored for the different needs in different projects and organizations. Depending on the type of each metric, we choose the appropriate mechanisms for data collection and validation. As goals, questions and metrics provide for tractability of the (top-down) definitional quantification process, they also provide for the interpretation context (bottom-up). This integration of definition with interpretation allows for the interpretation process to be tailored to the specific needs of an environment.

GENERATING A PARTICULAR OPERATIONAL MODEL

Often we must build simple models of various products and processes. For example, suppose we wanted to characterize the education and training of an individual team with regard to a particular process, e.g., a method or technique. We begin by trying to define the steps of our education and training process. For example, suppose we begin by providing the individual with training manuals and expecting them to be read. We then provide a course, educating the individual in the process. This is followed by training via an application of the process to a toy problem in order to build up skills in using the process. The individual is then assigned to a project that is using the process and receives some on the job training by a team member who is well versed in the application of the process. After this the individual is considered fully trained in the process.

This definiition is then converted into an operational model by providing a set of interval values associated with the various steps of the process. In this case, since the model is clear, each of the steps represents a further passage along the interval scale. Thus a value of 0 impies no training, 1 implies the individual has read the manuals, 2 implies the individual has been through a training course, 3 implies the individual has had experience in a laboratory environment, 4 implies the process had been used on a project before, under tutelage, and 5 implies the process has been used on several projects. Even though we call this a subjective rating, it should be clear that if the education and training process is valid, then our modle and the metrics associated with it are valid.

Using the GQM, we can generate a question that gathers the information for the model:

Characterize the process experience of the team.
(subjective rating per person)

- 0 - none
- 1 - have read the manuals
- 2 - have had a training course
- 3 - have had experience in a laboratory environment
- 4 - have used on a project before
- 5 - have used on several projects before
- x - no response

The data from the question can then be interpreted in a variety of ways. For example, if there are ten members of the team, we might requrie that a minimum requiremnt is that all team members have at least a three and the team leader has a five, etc. This evaluation process will become more effective over time.

A PROCESS GOAL EXAMPLE

As an example of a process goal, consider the following:

GQM Goal: Analyze the system test process for the purpose of evaluation with respect to defect slippage from the point of view of the corporation.

This is a simple example of a process evaluation goal. It should be noted that we may wish to mix process and product measurements on several occasions. For example, here we may wish to have several product measurements in the environment characteristics to help us locate similar projects in the data base for comparison.

This goal requires a model of the object: the system test process, and a model of the quality perspective of interest, defect slippage. A process model should have associated with it a procedure or set of steps to be performed, and a goal that provides information about how to evaluate whether the process has been effectively applied (process conformance). Since the purpose is evaluation, the focus model should have associated with it a model of the quality aspect of interest and an algorithm for interpreting the results. Consider the following two simple models:

System Test Process Model:

Process Goal: Generate a set of tests consistent with the complexity and importance of each requirement.

Process Procedure: (1) Enumerate the requirements, (2) Rate importance by marketing, (3) Rate complexity by system tester, (4) ...

Defect Slippage Model:

Let:
Es = #faults per KLOC found in system test in this project
Ea = #faults per KLOC found in acceptance test in this project
Eo = #faults per KLOC found in operation in this project

Let {Pi} be the set of projects used as a basis for comparison.
PEs = average #faults per KLOC found in system test in {Pi}
PEa = average #faults per KLOC found in acceptance test in {Pi}
PEo = average #faults per KLOC found in operation in {Pi}

Let **Fc** = the ratio of faults per KLOC found in system test to the faults found after system test on this project. [Fc = Es/(Es+Ea+Eo)]

Let **Fs** = the ratio of faults per KLOC found in system test to the faults found after system test in the set of projects used as a basis for comparison.
[Fs = PEs/(PEs+PEa+PEo)]

Let **QF = Fc/Fs** = the relationship of system test on this project to faults as compared to the average the appropriate basis set.

Simple Interpretation Algorithm for Defect Slippage Model

```
if QF
  > 1 then
          method better than history
          check process conformance
          if process conformance poor
                improve process or process conformance
          check domain conformance
          if domain conformance poor
                improve object or domain training

  = 1  then
          method equivalent to history
          if  cost lower than normal
                  method cost effective
                  check process conformance
          ...

  < 1  then
          check process conformance
          if process conformance good
                check domain  conformance
                if domain conformance  good
                        method poor for this class of project
```

This model can be used not only at this high level but for each class of fault possibly weighted by the total cost to isolate and fix a fault. We can do the calculation by error and failure category as well.

ENVIRONMENT: [The environment consists of the following: process factors, people factors, problem factors, methods, tools, constraints, etc. It consists of all the characterizing metrics that may not be directly relevant to the study but whose existence allow us to choose the appropriate "standard" project set used for comparison and may explain various differences.]

PROCESS QUESTIONS

Process Conformance:

A model of the process and models of assessment of how well each of the steps are performed

Characterize the test method experience of the test team.
(subjective rating per person)

0 - none
1 - have read the manuals
2 - have had a training course
3 - have had experience in a laboratory environment
4 - have used on a project before
5 - have used on several projects before
x - no response

[Note that these questions represent a subjective model of experience with the test method measured on an interval scale, i.e. 5 is better than 4, etc.]

How many requirements are there?
(enumerate them)

What is the importance of testing each requirement?
(Subjective rating 0 - 5 by marketing and testers)

0 - not important, could be left out
1 - not too important, may affect some users
2 - mildly important, will affect some users
3 - important, should affect most users
4 - extremely important, part of the essence of the system
5 - critical, without this the system is useless
x - don't know

What is the complexity of testing each requirement?
(subjective rating 0 - 5 by tester)

0 - doesn't need to be tested
1 - easy to test, one test should do it
2 - reasonably easy to test, only a few ad hoc tests are needed
3 - not easy to test, requires carefully made up test suite
4 - very difficult to test, requires a lot of thought to make up a good test suite
5 - extremely difficult to test, requires a large, complex test suite
x - impossible to test

What is the distribution of tests over requirements?
(number of tests/requirement)

Is the number of tests/requirement consistent with the evaluation of its complexity and importance?

0 - there are no tests for this requirement
1 - there is at least one test
2 - there are several tests but not nearly enough
3 - the number of tests are reasonable but not sufficient given the

importance or complexity of the requirement
4 - the number of tests are sufficient for the complexity and importance of the requirement
5 - the number of tests are more than adequate for the importance and complexity of the requirement
x - no response

[This may be calculated subjectively, as is done here or there may be a quantitative evaluation based upon some predefined expectation. For example:
<u>If</u> importance = 5 and complexity = 5
<u>then</u> <u>if</u> number of test cases =

Domain Conformance:
A quantitative characterization of the object to which the process is applied and an analysis of the process performer's knowledge concerning this object

How familiar is the domain?
(subject rating 0 - 5 for each tester)
0 - domain new to me
1 - have had a course in the subject domain
2 - have built or tested one system in this domain
3 - have built and tested at least one system in this product line
4 - have built and tested several systems in this domain
5 - have tested and built several systems in this product line

How understandable are the requirements?
(subjective rating 0 - 5 for each requirement)
0 - not understandable at all
1 - requirement ambiguous or not sure what it means
2 - not sure of the full ramifications
3 - reasonably clear requirement
4 - requirement is perfectly clear
5 - have successfully tested this type of requirement before

How precisely are the tests (inputs, results) known in advance? (subjective rating 0 - 5)
0 - there were no tests for this requirement
1 - will make the inputs up at terminal
2 - know the inputs but not the results
3 - know the inputs and the range of the results
4 - know the inputs and the results
5 - have simulation results for the test cases

How confident are you that the result is correct?
(subjective rating 0 - 5)
0 - there are no results
1 - the results are incorrect
2 - not sure the results are correct
3 - think they are correct
4 - reasonably sure they are correct
5 - positive they are correct

Are tests written/changed consistent with the evaluation of their complexity and importance?

(Subjective rating 0 - 5)

0- there are no tests for this requirement
1- there is a least one test
2- there are several tests but not nearly enough
3- the number of tests are reasonable but not sufficient given the importance or complexity of the requirement
4- the number of tests are sufficient for the complexity and importance of the requirement
5- the number of tests are more than adequate for the importance and complexity of this requirement

What is the evaluation of the domain conformance?

[This may be calculated subjectively, as it was done above or there may be a quantitative evaluation based upon some predefined expectation.]

Focus: ***Cost***
A quantitative specification of the costs of interest. This perspective is needed because it is used in the interpretation of the defect slippage model evaluation

What is the total cost of testing?

[Note that a more detailed level of granularity can be given for costing as shown below.]

What is the staff time to make a test?

What is the staff time to run a test and check the result?

What is the staff time to isolate the fault?

What is the staff time to design and implement a fix?

What is the staff time to retest?

What is the machine time used?

Focus: ***Defect Slippage Model***
A quantitative specification of the defect slippage quality perspective. The questions here relate to the data required by the defect slippage model

What is the number of faults failures discovered during system test, acceptance test and one month, six months, one year after system release on this project?

What is the number of faults failures discovered during system test, acceptance test and one month, six months, one year after system release on the set of projects classified as similar?

What is the ratio of faults in system test on this project to faults found from system test on?

What is the ratio of faults in system test on the set of similar projects to faults found from system test on?

What is the ratio of system test performance on this project to system test performance on the set of similar projects?

Focus: ***Various General Defect Slippage Model s***
[A quantitative specification of the quality perspective a various general defect slippage models. This data could be used to calculate any of a number of more detailed models.]

What is the number of errors, faults and failures on this project
    in total,
    per line of code,
    by various classification schemes, and
    by cost to isolate, fix and overall,
Discovered during each phase of development and one month, six months, one year after system release?

What is the number of errors, faults and failures on the set of similar projects
    in total,
    per line of code,
    by various classification schemes, and
    by cost to isolate, fix and overall,
discovered during each phase of development and one month, six months, one year after system release?

What is the ratio of weighted faults in system test on this project to faults found from system test by various classification schemes?

What is the ratio of weighted faults in system test on the set of similar projects to faults found from system test by various classification schemes?

What is the ratio of system test performance based upon the various slippage models on this project to system test performance on the set of similar projects?

FEEDBACK

This includes questions related to improving the process relative to the quality perspective of interest

Does the system test method need to be refined or modified?

Is more or different training needed in the method or the technology?

Is more or different training needed in the application domain?

What should be automated?

What is the input to the requirements, specification, design and code techniques, methods, and tools, and the defect detection techniques, methods, and tools?

DATA SOURCES

System test tables:

System test table 1: Nature of requirements (Filled out after baselining of requirements)

Req. #	How understandable is the requirement	Importance of testing (marketing)	(systest)
R1	5	3	2
.			
.			
.			

(Req# is the list of each system or component requirement, how understandable is the requirement is the subjective rating, importance of testing is a subjective rating by the marketing and system test group of how important the requirement is)

System test table 2: Nature of tests (Filled out after test plan)

Req. #	#of tests	How well are tests known?	Difficulty of testing	Evaluation subj	stat	Are # of tests consistent with diff & import?	Rating
R1	5	3	2	3	3/50	4	4
.							
.							
.							

(Req# is the list of each system or component requirement, # of tests is the number of tests run against that requirement. How well were the tests known in advance is a subjective rating, difficulty of testing is a rating by the test group of how hard they think it is to actually test the requirement sufficiently, evaluation consists of a subjective rating by the tester of the quality of the tests for that particular requirement, i.e. is the test set "adequate" for the requirement? and a statistical evaluation, performed by the software engineering group of the number of tests run (weighted by their subjective quality, their importance and the difficulty of testing) divided by the total number of tests run for all requirements. Rating is calculated by the SE group based upon the weighting of the question.)

System test table 3: Results of the tests (Filled out after tests run)

Test #	Failure? Yes : No	How confident are you in the results?	# of Faults found
T1	x	3	1
.			
.			
.			

(Req# is the list of each system or component requirement, How confident are you in the results? is a subjective rating, # of errors after test is filled out by the system test group based upon the number of errors associated with the particular requirement based upon the given set of test cases.)

Acceptance Test Table:

Defect | Defect Classifications |

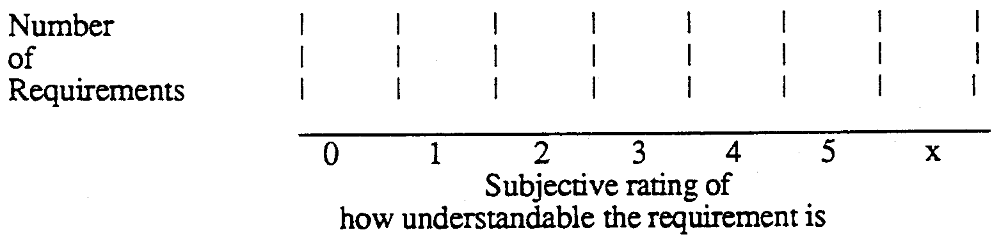
Operation Table:

Defect | Defect Classifications |

DATA PRESENTATIONS

Slippage model data:
QEs, REs, RPEs
Es, Ea, Eo
Histograms of:
Number of faults found in each phase
The number of requirements vs. subjective ratings of
how understandable the requirement is
importance of requirement
difficulty of testing the requirement
. . .

Example:



A PRODUCT GOAL EXAMPLE

As stated above there may be many perspectives taken on the same product. In this product example, we use the final product and take two different perspectives on it.

GQM Goal: Analyze the system for the purpose of evaluation with respect to reliability from the point of view of the user community and with respect to user satisfaction from the point of view of the customer.

Product Models: Logical/Physical Attributes, Cost, Changes, Operational Profile

PRODUCT QUESTIONS

Logical/Physical Attributes:

Classify the application domain and solution domain of the final product?

What is the function of the final product?

What is the size of the final product with respect to:
pages or lines of user documentation,
source lines with comments,
executable statements,
program units, e.g., functions, processes, components,
the number of requirements,
etc.?

What is the complexity of the product with respect to:
syntactic control (e.g., cyclomatic complexity, software science metrics),
data use (e.g., span, data bindings),
etc.
for each appropriate unit?

What is the distribution of programming language features used?

What are the run-time time and space constraints?

Cost

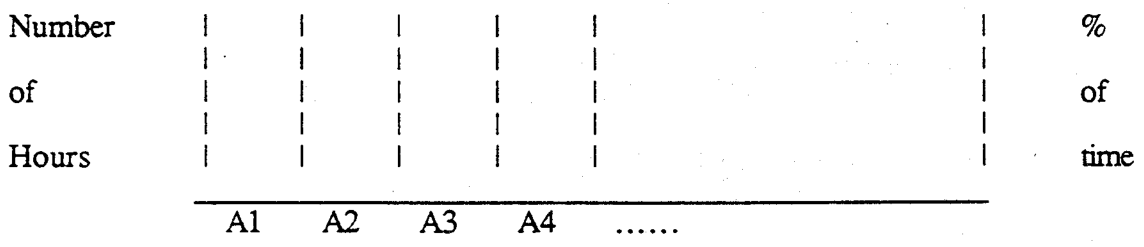
What is the effort by phase, activity, personnel type, used to develop the system?

How much machine time was used to develop the system; documentation as well as other aspects?

How much calendar time was used to develop the system and each component?

DATA PRESENTATIONS:

Histogram of:
the number of hours and the percent of time vs. the time card accounting sub-codes
the # of hours and the percent of time vs. the various activities (from the beginning of data collection and during the last report period).



Where A1, A2, A3, etc., are the different activities for which time is being accounted.

Changes

A. Enhancements

What is the number of enhancements (normalized by calendar time, phase of project)?

What are the enhancements categorized by type (requirements, specification, design, architecture, planned enhancements, insert/delete debug code, improve clarity, optimize: space or time, feature, enhancement, bug)?

What are the enhancements categorized by market/external and internal needs?

What are the changes characterized by size, e.g., number of lines of code, number of components affected, etc.?

What are the number of changes characterized by disposition, e.g., rejected as a change, not relevant, under consideration, being worked on, completed, saved for next enhancement?

What are the changes characterized by level of document changed?

How many customers are affected by the changes?

What is the Trouble Report history profile for each change?

B. Defects

What are the number of errors in total and by type, e.g., error origin, error domain?

What are the faults in total and by type, e.g., fault entry time, fault detection time/phase, (omission, commission), software aspect, mechanism of discovery, product level?

What are the failures in total and by type, e.g. severity, failure detection time/phase?

What are the number of errors, faults and failures in total and by type, normalized by phase and calendar time?

DATA PRESENTATIONS:

Histograms of the number of changes/defects by various classes.

Graphs of # of changes/defects initiated and closed vs. calendar time

Number of errors/component for each component

Context

A. Customer community

What classes of customers are expected to use the system?

What is the matrix of functional requirements vs. customer classes?

What is the matrix of functional requirements vs. components of the system? .

B. Operational profile

What percent of the system is expected to be executed by each customer class?

DATA PRESENTATIONS: Various Matrices

Requirements x Customer Matrix: For each requirement, the percent a customer class will make use of that requirement, i.e.,

	CUSTOMER
REQUIREMENTS	percent use by customer

Component x Requirements Matrices:
Class A: For each design and code component the percent of that requirement that is covered by that component
Class B: For each component, the percent that component covers a particular requirement

$$\begin{array}{l|l|}
 & \textbf{REQUIREMENTS} \\
\textbf{COMPONENTS} & \text{percent } R_i \text{ covered by } X_j \quad = 100 \\
 & \text{percent } X_i \text{ covers } R_j \quad >= 100
\end{array}$$

Component x Customer Matrix: For each component the percent use by a particular customer class

$$\begin{array}{l|l|}
 & \textbf{CUSTOMER} \\
\textbf{COMPONENTS} & \qquad\qquad \\
 &
\end{array}$$

Test x Component Matrix: For each test, the set of components it executes

$$\begin{array}{l|l|}
 & \textbf{COMPONENTS} \\
\textbf{TESTS} & \text{coverage} \\
 &
\end{array}$$

Test x Requirements Matrix: For each test, the set of requirements it covers

$$\begin{array}{l|l|}
 & \textbf{REQUIREMENTS} \\
\textbf{TESTS} & \text{coverage} \\
 &
\end{array}$$

Focus 1: Examine the <u>reliability</u> of the system from the <u>user's</u> point of view.

Major model used: some MTTF model (e.g., Musa)

What is the MTTF for the overall system, during system test, acceptance test, and for each customer base during operation?

What is the estimated fault density for the model? target MTTF?

DATA PRESENTATIONS:

Graphs of the MTTF vs. execution time and calendar time containing actual mean-time to failure data and projected mean-time to failure data

Validity of the model for the project:

How many requirements are there? What is the distribution of tests/requirement?

What is the customer/requirements matrix with probabilities (usage weightings) for each requirement?

How accurately does that describe each user operational profile?

Do the test cases reflect the customer/requirements probability matrix, i.e.,
is the test suite made up according to the different operational scenarios of the customer bases?

Is the probability distribution of test cases run during system and acceptance test based on customer OP profile?

Were the test cases randomized and distribution based on the customer operational profile from the previous question when they were run ?

Are corrections being made as failures are discovered?

Are new errors being introduced into the system during testing and debugging?

Is the model being run for each customer base with a different operational scenario?

Do the cumulative changes imply that the reliability model should be restarted?

Validity of data collected:

How valid is the failure data and the associated failure times?

Substantiation of the model:

Use of error profiles:

What is number of faults detected per 1000 lines and per component (one-half system test, system test, one-half certification, end of certification)?

Does this agree with previous projects' fault history and reliability ratings?

Use of coverage data:

What is the requirement, component coverage of the test plan at various points in time?

Do the requirement and component coverage correspond to the distribution in the (requirement, component) and (component, customer) matrices?

Focus 2: Examine the user satisfaction from the customer's point of view

Major model(s) used: User response to system
How many failures are reported by the users?
How many clarifications are requested by the user?
Is the use of the system growing, shrinking, staying the same?
How many requests are there for functional enhancements?
How many functional change requests are real versus functions already in the system?
How many performance change requests are being submitted?

Validity of the model for the project: User representation
Has a user committee been appointed that covers all user types?
Has the user community had any input into the requirements?
Are there prototype screens for the users to play with?
Is the user part of the test plan development?
Is the user part of the test team?

Validity of the data collected:
How valid is the data collected?

Substantiation of the model: User subjective evaluation
How responsive is the system to user request for functionality?
How responsive is the system to user request for performance?
How does the user rate the system with respect to:
Ease of use,
Functionality,
Performance
Ease of understanding the documentation?
(A questionnaire should be made up)
Why did you buy it?
Did you get more, less, same as expected?
How many others have you used?
Would you recommend it to a friend? A competitor?
Would you reorder?
Which competitors did you consider
How many requests are there for functional enhancements?

FUTURE DIRECTIONS

In summary, the Goal Question Metric Paradigm (GQM) is a mechanism for defining and interpreting operational and measurable software goals. It combines models of an object of study, e.g., a process, product, or any other experience model and one or more focuses, e.g., models aimed at viewing the object of study for particular characteristics that can be analyzed from a point of view, e.g., the perspective of the person needing the information, which orients the type of focus and when the interpretation/information is made available for any purpose, e.g., characterization, evaluation, prediction, motivation, improvement, which specifies the type of analysis necessary to generate a GQM model relative to a particular environment.

Although the GQM is being used today by several organizations, there is still a great deal of research and development needed to make it more effective and easier to use. As part of the TAME system [BaRo88], it needs to be automated. Full or partially automated support should be provided for creating goals, defining models, associating goals with models, generating questions that define the goals in terms of the models, generating the specifications for the data that needs to be collected, collecting and validating the data, analyzing the data in the context of the questions and models, interpreting the results with respect to the goals and an experience base for storing the results of the analysis and all the models and our history with the models. All these mechanisms must be coordinated and integrated to provide a complete and consistent view.

To support goal creation, it needs mechanisms to help in the selection and tailoring of existing goals based upon such factors as organizational needs, project requirements, past history and planning activities. One possible mechanism is a decision support system.

Work in model definition requires the development of various modeling languages for process models, product models and quality models. We need mechanisms to allow these models to be easily modified. Modification may take the form of generalizing, tailoring or allowing various options depending on various environmental factors.

The goal definition template needs to be automated to allow the instantiation of various models for any set of goals, the generation of questions that define the goals in terms of the models and the generation of the specification for the data that needs to be collected. This requires a formal basis for model selection and the definition of a goal generation language that automates the generation of questions and data based upon the goals and models.

Data can be collected automatically or via on-line forms, that can allow the collection, storage, and analysis of data based upon the models and the automated updating of the models in the experience based, from what we have learned in applying and evaluating the models.

Work in several of these areas is currently under investigation and in some cases prototypes exist.

References:

[Ba81a]
V. R. Basili, "Data Collection, Validation, and Analysis," in Tutorial on Models and Metrics for Software Management and Engineering, IEEE Catalog No. EHO-167-7, 1981, pp. 310-313.

[Ba85a]
V. R. Basili, "Quantitative Evaluation of Software Engineering Methodology," Proc. of the First Pan Pacific Computer Conference, Melbourne, Australia, September 1985 [also available as Technical Report, TR-1519, Dept. of Computer Science, University of Maryland, College Park, July 1985].

[Ba89]
V. R. Basili, "Software Development: A Paradigm for the Future", Proceedings, 13th Annual International Computer Software & Applications Conference (COMPSAC), Keynote Address, Orlando, FL, September 1989

[BA90b]
V. R. Basili, The Goal/Question/Metric Paradigm, white paper, University of Maryland, 1990.

[BaRo88]
V. R. Basili, H. D. Rombach "The TAME Project: Towards Improvement-Oriented

Software Environments," IEEE Transactions on Software Engineering, vol. SE-14, no. 6, June 1988, pp. 758-773.

[BaSe84]
V. R. Basili, R. W. Selby, Jr., "Data Collection and Analysis in Software Research and Management," Proc. of the American Statistical Association and Biomeasure Society Joint Statistical Meetings, Philadelphia, PA, August 13-16, 1984.

[BaWe84]
V. R. Basili, D. M. Weiss, "A Methodology for Collecting Valid Software Engineering Data," IEEE Transactions on Software Engineering, vol. SE-10, no.6, November 1984, pp. 728-738.

[BoBrLi76]
B. W. Boehm, J. R. Brown, and M. Lipow, "Quantitative Evaluation of Software Quality," Proceedings of the Second International Conference on Software Engineering, 1976, pp. 592-605.

[KoAk83]
M. Kogure, Y. Akao, "Quality Function Deployment and CWQC in Japan," Quality Progress, October 1983, pp.25-29.

[McRiWa77]
J. A. McCall, P. K. Richards, G. F. Walters, "Factors in Software Quality," RADC TR-77-369, 1977.

Appendix

It might be usefül to compare the other approaches to quality measurement in terms of the parameters of the GQM approach. The following figure offers a highlight of the differences among the approaches.

Comparison of Quality Measurement Approaches

Criteria	QFD Approach	SQM Approach	GQM Approach
Scope			
Object of study	products	final product	any process product, model
Purpose	plan, engineer, control	assess	characterize evaluate, predict, motivate, ...
Viewpoint	customer user	customer user	customer, user, developer, manager, corporation...
Structure			
Paradigm	Trace user characteristics of final product into related product/process characteristics at various stages of development	Refine factors into criteria and metrics	Refine goals into questions and metrics
Options	select/tailor	select	select/tailor
Usage	Quality Management	Quality Management	Quality and Project Management

The SQM approach was developed to allow the customer to assess the product being developed by a contractor. Thus, the object of the SQM evaluation is the final product for the purpose of assessment from the point of view of the customer.In this case a set of factors is defined on the final product, e.g. , which are refined into a set of criteria, which are further refined into a set of metrics. The models are defined and the user selects the particular set of factors and criteria of interest. It can be thought of as representing a specific example of a GQM with the models and metrics alreadv supplied.

The QFD approach was originally developed for manufacturing in order to better understand customer requirements and map them into the design documents for the product. The approach is being modified for software development. Thus the objects of study of the QFD are the various software documents for the purpose of planning via controlling and engineering the product to satisfy the customer needs. As with the SQM, the models and metrics are built into the system and supplied to the user although there is some opportunity to tailor. Again, this can be considered as a special example of the GQM.