

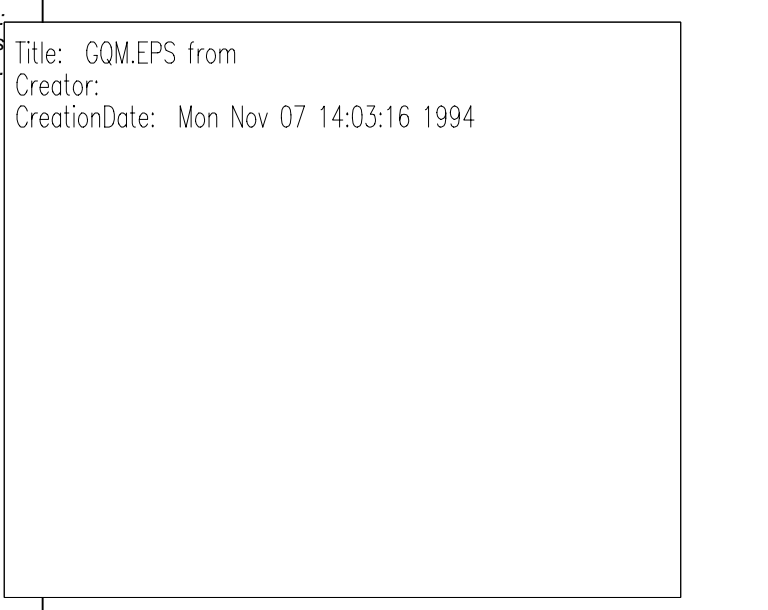
Creating a Metrics Program

Step 3: Define Metrics Required to Reach Goals

Step 3 is to choose which software metrics will help achieve the goals of the metric program.

The goals for a metrics program determine the specific metrics on which the program will be based. This section analyzes each of the eight goals outlined in Step 2. The goals will be converted into a set of metrics by applying the Basili GQM (Goals/Questions/Metrics) model (*Basili, 1984*). The following diagram illustrates the Basili model.

Figure 3.2 :
Basili process
diagram.



Title: GQM.EPS from
Creator:
CreationDate: Mon Nov 07 14:03:16 1994

The GQM model helps you identify a set of metrics which support your goals. This is done in 2 steps:

1. Determine a set of questions which, when answered, provide the insight necessary to achieve the goals.
2. Determine a set of metrics which can be collected and analyzed to help answer each question.

You should follow this procedure for each goal you selected in Step 2. The result will be a set of metrics which support your goals.

If you selected goals directly from the list in step 2, questions and metrics will be provided for you in the following sections.

Keep in mind that the metrics defined at this stage are quite simple. They are going to be used to help get the program going and to establish a baseline. Later, when the mechanics of the metrics program have been worked out, more complicated or specialized metrics can be added. It can be tempting to go overboard and measure everything right away, but this leads to information overload. Attempt to focus on metrics which will help you achieve your initial goals.

Questions and Metrics for the Program Goals

In this section, questions are presented for the goals from step 2, followed by tables that list metrics which support each question.

Some software attributes can be measured in various ways (e.g., size can be measured in lines of code, function points, tokens etc.), and there are no industry wide standards governing which metric to use. The metrics used in SPC's Metricate database were selected because they are appropriate to an initial metrics program, and helpful in illustrating the creation of a metrics program. These include the following:

*Figure 3.3 :
Standard
metrics
selected for this
book.*

Attribute	Selected Metric
size	SLOC
effort	person hours (or PH)
cost	dollars
complexity	software category

These metrics can be used without modification for most projects. If you are more comfortable with other metrics, substitute as appropriate. The SPC's Metricate procedures can be applied equally well to other metrics.

Development activities are comprised of the work units necessary for their completion. Depending on the activity, a unit may be a document, a test or a module of code.

This documentation uses many other terms consistently throughout the book. Definitions of many terms, including those listed above, are listed in the Glossary section. Please refer to the Glossary to become familiar with these terms, if necessary.

Questions for Goal 1: Improve Development Process

Improving the software development process is the best all-round way of reducing development costs while maintaining overall product quality. Even if you do not explicitly choose this as one of your metrics goals, you will probably make changes to the development process in order to accomplish other goals.

Some questions to ask about the development process:

1. *How well does the development process describe the work being performed?*

We like to think of the software development process as an all-encompassing set of rules that should be followed by everyone in all situations. Unfortunately, this is rarely the case. Most projects have some tasks which are performed in a manner that is inconsistent with the development process. One solution to this problem is to document the deviations, and then (if possible) adapt the process model to accommodate them.

One way to document process deviations is to require that staff members prepare a Process Exception Report (PER) each time they need to deviate. The goal is to minimize or eliminate PERs through strategic process modifications.

2. *What is the relative effort for each activity in the process?*

The relative effort spent on each process activity should reflect the expectations of the development process. Sometimes a process is implemented that simply does not reflect the work being done. You can identify this by comparing the relative effort performed in each activity to the relative effort expected. Often seeing where time is actually spent will highlight process deficiencies.

3. *What is the elapsed time for each activity in the process?*

Are some activities consistently taking longer than they should? This could be caused by many things, but to detect it, you must measure the process over a number of projects for trends to be established.

4. *Where in the process are defects being introduced? detected? corrected?*

These questions are extremely important. Defects have a negative effect on the development effort as well as on the schedule.

A good process will detect errors early and correct them promptly. Of course, the overall objective is to minimize defects.

5. *How many requirements are added during the process?*

If the requirements are explicitly stated at the start of the project, there should be minimal changes to them during the project. Changing requirements during later activities can be very costly, as the changes affect all activities. A late change could require that staff redo work performed in previously completed activities. It is desirable to document and sign off all the requirements at the earliest possible date.

Figure 3.4 lists some metrics that will be useful in researching the above questions:

*Figure 3.4 :
Process
improvement
metrics.*

Goal 1:	Metrics
Improve development process	<ul style="list-style-type: none"> • Average elapsed time between defect identification and correction • Number of person hours (effort) to complete each activity • Elapsed time for each activity • Number of defects detected in each activity • Number of deviations from the software development process • Number of requirements added or changed during development

Questions for Goal 2: Improve Software Estimation

Reliable estimates of software size, schedule, and development cost are vital to a successful project. There are many estimation strategies available to assist in creating accurate estimates (see Appendix C - Software Estimation for an overview of the subject).

One of the best techniques for obtaining accurate estimates is to use company baseline data, or data from previous projects. Refer to the section "Activity-Based Cost Estimation" in Appendix C for additional information.

The following questions should help determine which metrics to collect:

1. *What is the actual versus estimated labor rate for each activity?*

Each activity in the software development process has a labor rate associated with it. This labor rate is an expression of the level of effort required to perform the activity, and is measured in PH/SLOC (person hours per line of source code).

This value is required for all estimating, and if it is not currently tracked, you must make an informed guess as to what it is. Once you have tracked this value through a few projects, it will be easier to estimate.

2. *How much have the requirements changed since the initial estimates were made?*

Unstable requirements can have a serious impact on the estimates of the work to be performed. To improve the estimates, you should have some idea of how much the requirements typically change, and how the changes affect the initial estimates.

3. *How complicated is the software being developed?*

The labor rate may vary according to the complexity of the software being developed. A simple method of estimating complexity is to categorize the software based on its type. Some possible software types include the following:

- batch MIS software
- interactive MIS software
- embedded software
- database applications
- data communications

- graphical user interface (GUI) development
- process control software
- systems software
- scientific applications
- I/O subsystems
- military systems
- visual programming software (generated GUI)

Add any other software types that are appropriate.

Each type of software will have its own labor rates, and therefore different development costs.

4. *What is the actual versus estimated schedule, effort, and size for each activity?*

To verify the degree of accuracy of the estimates, monitor the actual schedule, effort, and size. If the estimates are consistently high or low, adjust them accordingly.

5. *What is the actual versus estimated staffing level? overtime worked?*

It is possible to deliver software on schedule but be significantly over budget. This could be due to increased staffing levels or to overtime. Tracking of overtime is essential if you want to use your baseline data to accurately estimate effort for future projects.

Figure 3.5 lists the metrics to use for answering the questions above:

Figure 3.5 :
Software
estimation
metrics.

Goal 2:	Metrics
Improve software estimation	<ul style="list-style-type: none">• Initial estimate versus actual effort (person hours) for each activity• Initial estimate versus actual project schedule for each activity• Initial estimate versus actual size of the software (new and reused)• Initial estimate of staff required versus actual staff levels (for each activity)• Total overtime hours• Labor rate (PH/SLOC) for each activity• Requirements changed for each activity• Software product complexity

Questions for Goal 3: Improve Project Tracking

The ability to accurately track the status of projects is an essential component of successful software project management.

Project tracking metrics are slightly different from other metrics in that they are dynamic: the values change as the project progresses (e.g., percent of work complete). At a given time, these metrics represent a snapshot of the state of development.

Use these metrics by comparing them against their corresponding estimates (in the areas of schedule, size, and cost). If there are discrepancies, you can take corrective action before it is too late.

Here are some questions that are useful in determining the status of a project:

1. *What is the status of each development activity?*

The reason projects are tracked is to identify deviations from the estimated cost or schedule. Track the status of each activity on an ongoing basis. The status should include information on the effort, staffing, elapsed time, and proportion complete. Investigate deviations so you can update the estimates.

2. *What is the status of the overall project?*

Using the status information collected for each activity, you should be able to assess the overall status of the project, to ensure that it is in agreement with the company objectives and/or the contract.

3. *What is the earned value of each activity?*

Earned value is a measurement of the actual amount of work accomplished. It is the best all-round indicator of the degree of completeness of the project (and of each activity in the project). Calculate the earned value to identify activities which are not progressing, and to revise the labor rate and/or schedule. See Appendix D - Metrics Formulas.

4. *How do actual project expenditures compare to the budget?*

There are many people who are concerned with the bottom line: the cost. Track the project expenditures against the estimated budget to ensure that extra expenses are not incurred.

Figure 3.6 summarizes the project tracking metrics:

Figure 3.6 :
Project tracking
metrics.

Goal 3:	Metrics
Improve project tracking	<ul style="list-style-type: none">• Earned value of each activity• SLOC completed• Initial estimate for SLOC• Overall percent of work complete• Percent of work complete for each activity• Percent of budget spent to date• Percent of schedule elapsed• Proportion of tests executed• Proportion of tests passed

Questions for Goal 4: Minimize Schedule

The project schedule is probably the source of more headaches than any other component of software development projects. Most developers and managers do not appear to have a good idea of how their company's time is actually spent.

By collecting metrics related to how people actually work and then relating them to the schedule, you can gain a greater understanding of how time is being used. Eventually, you can target sections of the schedule for improvement, either through process enhancements or staff or budget changes.

Here are some questions that will help in this process:

1. *What is the actual schedule for each activity?*

To minimize the schedule, you must know how time is currently being spent. Start by measuring actual schedules for each activity.

2. *What is the actual level of effort for each activity?*

Look at the level of effort expended for an activity to understand the actual schedule for the activity.

3. *How much time has been spent on rework?*

Rework is the time spent fixing problems. Problems can be with code, design, requirements, or any other output from an activity in the development process. Not only is rework expensive, it is also costly in terms of schedule. To minimize the schedule, the number of hours spent on rework must be minimized. The best way to do this is to find and fix problems early in the development cycle.

4. *How much time has been spent on non-development tasks?*

Non-development tasks include the support, maintenance, and administrative duties that do not constitute a part of the product being developed. If too much time is spent on these activities, you are wasting time that could be spent developing the product. If not enough time is spent, the product could get bogged down when the product developers have to take on support roles themselves. By measuring the level of non-development tasks across a number of projects, you can determine exactly how much non-development time is best for the company.

5. *How many overtime hours have been required in each activity?*

Excessive overtime indicates that the effort was underestimated or that the project is understaffed. Either case presents a problem in trying to minimize the schedule. Overtime hours are generally less productive than regular hours. In most cases it is better to increase staffing levels and minimize overtime.

6. *Is the staffing level adequate to meet the schedule as predicted?*

The initial schedule was based on a staffing level. The staffing level needs to be measured throughout the project to ensure that it is not too low. The schedule will be affected if the number of staff is not the same as the initial estimate.

Figure 3.7 summarizes the metrics related to the software schedule:

Figure 3.7 :
Software
schedule
metrics.

Goal 4:	Metrics
Minimize schedule	<ul style="list-style-type: none"> • Elapsed time between project milestones or activities • Initial estimate versus actual effort for each activity • Initial project schedule versus actual schedule • Initial estimate versus actual staffing levels • Person hours spent on rework • Total overtime hours in each activity

Questions for Goal 5: Minimize Development Cost

The development cost is the amount of money (in dollars) that is required to complete the project. Obviously, it is desirable to minimize cost.

Cost is often confused with effort. Effort is the number of person hours required to develop the project and is by far the largest component of the project cost. It is also the component that is most subject to change. Other components are the non-labor costs such as facilities, development equipment, and office supplies.

Here are some questions to answer. The answers should help you determine where the most money is being spent, and what improvements will help to minimize the cost.

1. *What is the cost in dollars of each activity?*

How much does each activity actually cost? The calculation of cost will vary from company to company, but the biggest portion will be the actual number of person hours required to complete the activity.

2. *What is the labor rate for each activity?*

The labor rate is measured in person hours per source line of code (PH/SLOC). Accurate values for labor rates are essential in determining the effort required for an activity. Store the effective labor rate for all projects in the metrics database for use in establishing standard company rates.

3. *What is the original versus the actual effort required for each activity?*

Software cost estimating depends on reliable estimates of effort. The best method for obtaining these estimates is to use comparable data from previous projects. Be aware of activities which demonstrate major discrepancies between actual effort and estimated effort, as they can lead to cost overruns.

4. *What is the estimated versus the actual cost for each activity?*

The initial cost estimates are the basis for allocating the available development funds (the budget) to each activity at the start of the project. It is useful to compare the final cost for each activity to the initial cost estimates to determine which activities are incurring the largest additional expenses.

5. *How much of the budget is spent on development versus managerial versus support tasks?*

Breaking the budget down by various categories of employees may indicate areas where costs could be saved. Try to determine a standard ratio of development to management to support staff for the company. Base the standard on the ratio which gives the highest overall productivity rate.

6. *How much of the budget is being spent to correct defects?*

If the amount spent doing rework is high, it is an obvious place to cut costs. The next step would be to determine if there are too many defects being produced, or if the cost of fixing defects is too high. Try to determine the costs of the defects and compare them across projects.

Figure 3.8 summarizes the metrics used to answer the questions about development costs:

Figure 3.8 :
Software cost
metrics.

Goal 5:	Metrics
Minimize development cost	<ul style="list-style-type: none"> • Actual cost for each activity • Amount spent fixing defects in each activity • Cost for each SLOC • Initial cost estimate for each activity • Budget for each activity • Initial estimate versus actual effort for each activity • Labor rate (PH/SLOC) for each activity • Percent of budget spent on development tasks • Percent of budget spent on management tasks • Percent of budget spent on support tasks

Questions for Goal 6: Improve Software Quality

Measuring software quality is a highly subjective task, since software quality tends to be difficult to define.

The question of defining software quality has been the subject of much discussion within software development circles. One popular approach is to look at the "ilities" of the system, or the attributes such as stability, maintainability, extendibility, verifiability, reliability, or portability.

Another good indicator of software quality is program correctness, which is often determined by looking at the average number of defects per size unit of code (e.g., defects/SLOC). This is referred to as the defect rate or defect density.

Expanding this further, the number of defects that remain in a piece of code when it is shipped can be estimated. This is referred to as the latent defect rate. This measurement is very useful, but is probably better covered in the advanced metrics texts listed in the References section of this document.

With this in mind, ask the following questions concerning the software quality:

1. *How many defects are there in the product?*

This is the main indicator of product quality. Determine how many defects were detected, how many were corrected, and how many remain. Also, track the types of defects identified and the development activity in which they were detected.

2. *Is the software maintainable?*

Maintainable software is critical to keeping expenses in control once the software has been released. Try to assess the degree of difficulty involved in updating the software. In particular, there should be an appropriate amount of inline documentation (LOD) for the size of the software.

3. *Is the software stable?*

Instability is characterized by software that gets into an inoperable state requiring system reset or similar actions. Determine the stability of the system by measuring the frequency with which the software becomes inoperable.

4. *Has the software been verified? Is it correct?*

Verified software is software which has been subjected to formal verification. Code walkthroughs and inspections are common mechanisms for verifying software and for detecting defects. Testing at different levels (such as functional, unit and integration testing) is also essential for verifying that the software is correct.

Figure 3.9 summarizes the metrics to use for improving software quality:

Figure 3.9 :
Software quality
metrics.

Goal 6:	Metrics
Improve software quality	<ul style="list-style-type: none"> • Average person hours to fix a defect • Mean time between failures (if appropriate) • Number of defects detected of each type • Number of defects/SLOC • Total lines of documentation (LOD) • Percent of code inspected

Note: Remember that terms such as defect type, inspected code, or LOD are defined more fully in the Glossary.

Questions for Goal 7: Improve Software Performance

Metrics are commonly used to measure the operational characteristics of software systems which, more often than not, relate to software performance.

The characteristics of a software system that define its performance vary substantially depending on the type of product being developed.

The following list of questions are general enough to apply to most types of applications. You can augment this question list with performance-related questions that are pertinent to the software you are developing.

1. *What is the processor utilization?*

Whether developing software for older processors or developing CPU-intensive applications, the overall processor utilization will be an important metric. Most modern target platforms come with tools that output the current processor utilization. Determining which pieces of the software are using the most cycles on the CPU may be another interesting measurement.

2. *What is the memory utilization?*

Some types of software must be developed with certain memory constraints in mind. If this is the case, collect the metrics indicating the memory usage.

3. *How is the software I/O performance?*

I/O operations can be between a screen or window and a user, or between the software and other software systems, or between the software and hardware. Most requirements will specify acceptable I/O processing levels which you can verify with metrics.

4. *What are the characteristics of the software?*

Different types of software have different operating characteristics. It is important to consider items such as the size, complexity, or stability of the software when attempting to assess overall performance.

Figure 3.10 summarizes the metrics to use to answer these questions:

Figure 3.10 :
Software
performance
metrics.

Goal 7:	Metrics
Improve software performance	<ul style="list-style-type: none">• Average CPU utilization• Average memory utilization• Mean time between failures (if appropriate)• Number of I/O transactions per unit of time (actual versus required)• Number of lines of code (SLOC)• Software product complexity

Questions for Goal 8: Improve Productivity

Productivity measurements are simply an extension of the cost and tracking metrics described earlier. Productivity is typically defined as the amount of work that can be performed in a unit of time. In software development this might be represented as the number of source lines that can be coded by one person in one hour (SLOC/PH). Note that this is the inverse of the labor rate (PH/SLOC).

Stabilizing or improving productivity is necessary in order to control staffing levels. It is also desirable to collect historical information on productivity measures to improve software estimation efficiency (Step 3, Goal 2).

Analyzing productivity involves looking at the software development process to find areas where effort is being expended unnecessarily.

Consider the following questions:

1. *How much time is being spent on rework?*

Excessive rework is a major detractor from productivity. By monitoring the amount of time spent on rework over a number of projects, you can determine a company average for rework time. Once this is established, you can ascertain the effects of rework on a specific project's productivity.

2. *Are developers spending too much time on support and managerial activities?*

If a project is understaffed in management and/or support positions, the developers will need to perform these tasks to keep the project moving. The more this happens, the more overall productivity will decrease. By maintaining data on the type of work performed by development staff, you should be able to identify optimal management and support staff levels.

3. *What is the average productivity?*

Productivity is measured in units of software produced per unit of time. This will be SLOC/PH if you use the conventions laid out in SPC's Metricate . The first step in improving productivity is to determine the current level. Measure the productivity of each of the activities in the development process.

4. *Is the productivity consistent with the experience of the team members?*

Productivity should increase as the level of staff experience increases. To interpret the productivity for a project, obtain a breakdown of the experience levels of project staff members.

5. *Are tools available to designers and managers?*

Automated tools, when used correctly, can increase overall productivity. Look at the availability and use of project management tools, development tools, and test tools.

Here are the metrics to use to answer these questions about productivity:

Figure 3.11 :
Productivity
metrics.

Goal 8:	Metrics
Improve productivity	<ul style="list-style-type: none"> • Average number of person hours spent on rework per development staff member • SLOC/person hours for each activity • Number of staff at each experience level • Percent of budget available for software development tools • Percent of budget available for support staff • Proportion of person hours spent on managerial or support tasks for each activity • Ratio of development staff per manager

Actions Required for Step 3

1. For each goal selected from the eight in Step 2, refer to the tables in the previous section "Questions and Metrics for the Program Goals", and determine which questions and metrics are appropriate for your company. Document the questions and metrics selected.
 2. If other goals were added in Step 2, devise a set of questions to help reach the goals. Use the questions to define a set of useful metrics, and document this information.
-

Step 3 Example

Metrics

In the previous step, you established that Buddicorp's goals were to improve its project tracking and estimating capabilities. In the discussions, you also learned that regardless of the goals identified and the process changes made, product quality was not to be adversely affected.

Consulting the sections in Step 3 that refer to your stated goals, you come up with the following list of questions for your metrics program:

1. *What is the status of each development activity?*
2. *What is the status of the project?*
3. *What is the earned value of each activity?*
4. *How do project expenditures compare to the budget?*
5. *What is the actual versus estimated labor rate for each activity?*
6. *How much have the requirements changed since the initial estimates were made?*
7. *How complicated is the software being developed?*

8. *What is the actual versus estimated schedule, effort, and size for each activity?*

9. *What is the actual versus estimated staff level? Overtime worked?*

10. *How many defects are there in the product?*

You add question 10 to facilitate tracking of overall product quality.

Using these questions and their related tables, you come up with the following set of metrics. These are the metrics that you will initially collect and analyze as part of your metrics program.

- Earned value of each activity
- Initial estimate versus actual project schedule for each activity
- Initial estimate versus actual effort (person hours) for each activity
- Initial estimate versus actual size of the software (new and reused)
- Initial estimate versus actual staff level (for each activity)
- SLOC completed
- Labor rate (PH/SLOC)
- Number of defects/SLOC
- Number of defects of each type
- Initial estimate for SLOC
- Overall percent of work complete
- Percent of work complete for each activity
- Percent of budget spent to date
- Percent of schedule elapsed
- Proportion of tests executed
- Proportion of tests passed
- Requirements changed for each activity
- Software product complexity
- Total overtime hours

