# Chapter 7: System Measurement Techniques

Measurement is a key element in the management of successful IT projects in every established engineering discipline. The objective of measurement is to provide IT project managers with the system information required to make informed decisions that influence project cost, schedule, and technical objectives. This chapter covers project measurement methods and tools.

## Project Measurement

Project measurement is a discipline that helps IT projects stay on schedule. Project measurement is a management tool that the project manager uses to identify risks, track specific problems, and assess the influence of these problems on project cost, schedule, and performance objectives. Project measurement provides accurate data to help the manager monitor the project's progress against the plan. The benefits of IT project measurement are as follows:

- ***Enhances effective communication.*** Communication among the members of the project is vital for its success. A well−known saying applies to communication in the IT industry: 'In the kingdom of the blind, the one−eyed man is king.' It is good practice to establish project measurement and share the information among the members of the team to avoid such an environment. This process establishes effective communication throughout the project and reduces the ambiguity that often surrounds many issues on a project. Measurement allows such issues to be explicitly identified, prioritized, tracked, and communicated at all levels of the project.
- ***Corrects problems.*** Correction of problems early in the project saves time and cost. Measurement focuses attention on the early discovery and correction of technical and management problems that can be more difficult to address later. The manager identifies potential problems early as risks that should be assessed and managed.
- ***Assists in tradeoffs.*** The manager decides tradeoffs on the basis of the project measurement data. Cost, schedule, quality, functionality, and performance all have to be managed together to make the project a success. A decision in one area often has an influence on decisions in other areas. Measurement allows the project manager to objectively assess these influences and make the proper tradeoff decisions to best meet project objectives. Even in highly constrained project environments, measurement helps the manager optimize performance within the scope of project objectives.
- ***Tracks project activities.*** Project tracking accurately describes the status of the project and represents the progress of the project activities. It helps the manager answer key questions, such as 'Is the project on schedule?' and 'Is the project product ready to deliver?'
- ***Helps managers make correct decisions.*** Making decisions is the primary duty of an IT project manager. The more accurate the measurement, the more accurate the data and the more correct the project management decisions. Measurement provides an effective rationale for selecting the best alternative.

Measurement establishes a basis for objective communication within the project team. This process helps the manager make decisions that materially influence the outcome of a project quickly and correctly. Measurement provides a baseline quantitative process for implementing risk management and financial performance on a project (Figure 7−1).

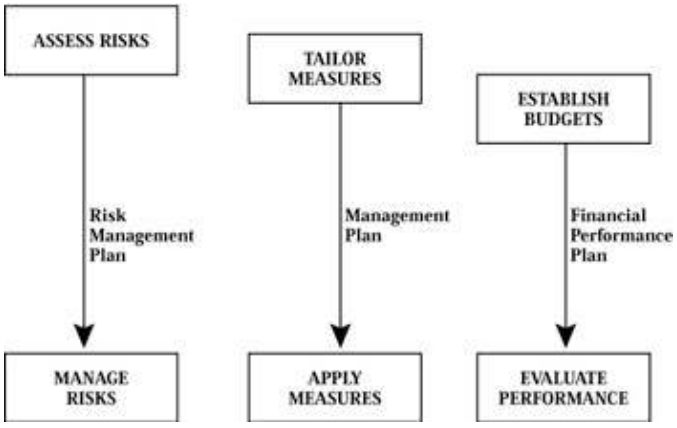



Figure 7−1: Quantitative project measurement process

## Project Measurement Metrics

Project measurement metrics is an excellent method of keeping track of the project's progress, various activities, performance, and expenditures. Metrics have been practiced in the IT industry for many years. In fact, people use measurements and metrics almost every day (e.g., driving a car, budgeting earnings and expenses, and preparing a list of necessities before going to the market).

If the manager cannot measure the progress of a project, he or she cannot manage it successfully. A successful project has the following factors:

- Goals
- Time
- Milestones
- Cost
- Budget
- Testing

A successful project 5 Plans 1 Budget 1 Schedule 1 Manpower. Project measurement plans consist of metrics that keep track of the project's progress (Figure 7−2). The following are characteristics of these metrics:

- Metrics without management discipline are not accurate metrics.
- A major factor of a successful project is an appropriate budget. Metrics showing time and cost accurately reflect budget.
- The project budget is directly correlated to the requirements. The project manager estimates project cost after analyzing and understanding the project requirements.
- Cost metrics help the manager spot problem areas.
- Cost saving is time divided by milestones. Each milestone defines tasks to be completed by that point. To keep projects on schedule, managers trade requirements for time.

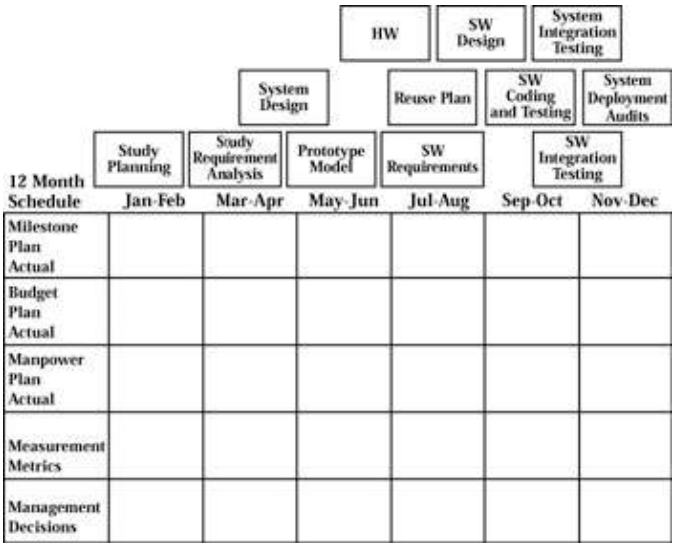


Figure 7−2: Project measurement process

The manager calculates typical metrics for the progress of manpower, hardware defects, software bugs, and travel and training expenses. A project without testing will be unsuccessful. Testing metrics help the manager verify and validate requirements and the correctness of the product. Testing metrics provide the manager with continuous feedback for the correctness of the project. Monitoring the project's progress leads to corrective action. The project manager must measure and pay attention to what the data are revealing. Case Study 7−1 provides a typical measurement example.

## Case Study 7−1

In the early 1990s, Debra Domeyer inherited a financial application development project in which the project team was extremely busy accomplishing little. The project, which was at a mortgage consolidation company, was rudderless, lacking a solid project plan, an authoritative project manager, and good tracking measures. More than a dozen people were at the first planning meeting. Domeyer attended, but there was no clear decision maker, and the project had a marathon scope.

'It had become a financial system that would also brush your teeth and make your bed in the morning,' says Domeyer, who is now Chief Information Officer (CIO) of PG&E Energy Services in San Francisco, California.

After the fateful first meeting, Domeyer estimated that the project was only 5% complete after 5 months of work. Given the anticipated spending rate and delivery speed, the new financial system would cost more than $1 million and take at least 2 years to complete. She reviewed the numbers with her Chief Financial Officer (CFO) and other senior executives, who quickly agreed to cancel the project. Domeyer then launched a new project with strict time, cost, and function requirements. She installed upgrades to the existing financial software package and met the most pressing business needs in 4 months−all for roughly $140,000.

A good project manager is the most important tool for successful delivery of a complex IT project, but even the most knowledgeable manager needs more than experience and gut feelings to close intricate IT implementation projects successfully. The manager can use a metrics discipline of instituting measurement programs to keep projects on schedule. Metrics can track everything from programmer productivity to business value.

Although it seems like common sense to measure a complex IT project's progress systematically, most CIOs bury metrics beneath all of the other demands on their time. In a 1999 study of IT managers in 6000 companies spanning major industries in 28 nations, Rubin Systems Inc., a consultant in Pound Ridge, New York, found that only 50% of the respondents actively use metrics. Of these, only 4% collect and use them on a weekly basis, 10% use them monthly, and 1% use them once or twice a year (Radosevich, 1999a).

## Management Measurement Criteria

The following are management measurement criteria for the success of an IT project:

- Schedule and progress regarding work completion: cost, schedule, and development progress (Figure 7−3)

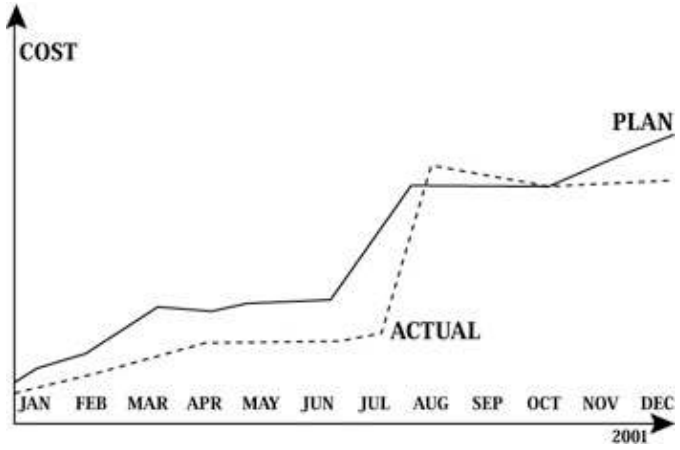
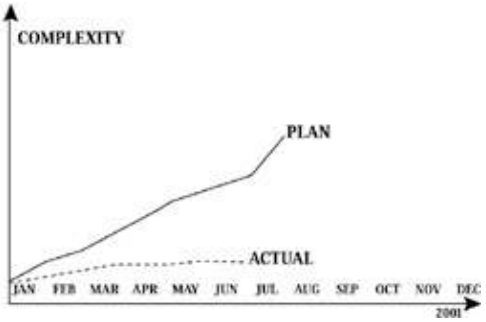


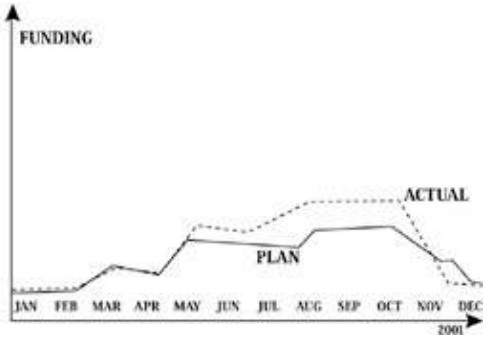

Figure 7−3: Sample cost, schedule, and project progress

- Growth and stability regarding delivery of the required capability: requirement traceability, requirement stability, design stability, development progress, and complexity (Figure 7−4, *A*)




7−4 A: Sample cost, schedule, and project progress

- Funding and personnel resources regarding the work to be performed: cost and manpower (Figure 7−4, *B*)




7−4 B: Sample funding and personnel resources

- Software development performance regarding the capabilities to meet program needs: software engineering environment
- Technical adequacy regarding software reuse, language, and use of standard data elements: computer resource use

The purpose for each of these metrics is as follows:

1. ***Cost.*** Tracks software expenditures (dollars spent versus dollars allocated)
2. ***Schedule.*** Tracks progress versus schedule (event/deliverable progress)
3. ***Computer resource use.*** Tracks planned resource capacity versus actual size (percentage of resource capacity used)
4. ***Software engineering environment.*** Rates developer's environment (developer's resources and software development process maturity)
5. ***Manpower.*** Indicates the developer's application of human resources to the development program and the developer's ability to maintain sufficient staffing to complete the project
6. ***Development progress.*** Indicates the degree of completeness of the software development effort; can also be used to judge readiness to proceed to the next stage of software development
7. ***Requirement traceability.*** Tracks requirements to code (percentage of requirements traced to design, code, and test cases)
8. ***Requirement stability.*** Tracks changes to requirements (user and developer requirement changes and their effects)
9. ***Complexity.*** Assesses code quality
10. ***Breadth of testing.*** Tracks testing of requirements (percentage of functions and requirements demonstrated)
11. ***Depth of testing.*** Tracks testing of code (degree of testing)
12. ***Fault profiles.*** Tracks open versus closed anomalies (total faults, total number of faults resolved, and amount of time faults are open by priority)
13. ***Reliability.*** Monitors potential downtime (software's contribution to mission failure)
14. ***Design stability.*** Tracks design changes and effects (changes to design, percentage of design completion)
15. ***Product quality.*** Regards the delivered products (fault profiles, reliability, breadth of testing, complexity, and depth of testing)

## Best Practices System Metrics

Best practices system metrics are calculated or composite indicators based on two or more measures. These metrics are used for measuring the progress of a project, improving a process, selecting methods and tools, and improving communication among practitioners. Comparison of this metric with a previous project provides a baseline for the data and gives meaning to the metric. Case Study 7−2 provides a typical metrics

example.

## Case Study 7−2

In 1997, when Jim Harding joined Olsten Corporation as a CIO, the $4.6 billion staffing and health services company had suffered several false starts trying to implement a nationwide, integrated IT system. The abandoned projects had cost roughly $5 million, years of wasted time, and the information system (IS) department's credibility.

Harding was not taking chances. He knew he had to put detailed measurement programs in place to keep the lid on the $20 million project. The company's existing systems traced their origins back to 1969, when Olsten was a $100 million company and a single mainframe held sway over the data center. Since then Olsten has added a few IBM AS/400 computers to service divisional office support functions but not much else. Harding would have to start from scratch, adding all new PC−based local area networks (LANs), remote e−mail, enterprise resource planning systems, and a data warehouse in 30 months.

By 1997, Olsten was attracting major customers that wanted national and global contracts with such features as discounts, customized reporting, and consolidated invoicing. Harding's task was to provide systems that would let Olsten better manage its own burgeoning base of 80,000 contract workers and provide customers with global account management.

To keep his renewal project from imploding, Harding developed a project metrics program that tracked the following:

- ***Milestone.*** Timing the major chunks of the project, such as the LAN installation; project team members tracked hundreds of subevents leading to each milestone
- ***Performance testing.*** Simulating the heavy demand that Olsten's employees would put on the applications and networks
- ***Data conversion.*** Identifying data that had to be transferred to the new systems and writing programs to convert old to new
- ***Cost.*** Tracking expenses against original budget projections

The measurement program alone cost roughly $2 million, or 10% of the total cost of the project. However, with the company's entire IS system hanging in the balance, the metrics illuminated the difference between victory and ignoble defeat.

The project had its ups and downs, and the metrics played a role in both. For instance, with roughly 2 months to go before the company's new financial system was set to begin, Harding's data conversion spreadsheet showed that his team would need 2 weeks to load the new data, bringing the system down while they did it. The project team squeezed the conversion time down to 4 days by working with business leaders to purge old data and reduce the data they needed to convert and by tuning the financial application software. 'If we had not been able to reduce [the conversion time], we would not have been able to implement,' Harding says.

Achieving the milestones led to success. For instance, after installing the PC−based LANs, Olsten employees had a company−wide Intranet and e−mail system for the first time.

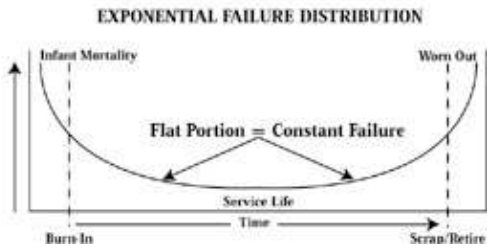
Olsten announced successful completion of its new system in June of 1999. Based on his experience, Harding advises devising metrics to catch mistakes and create smaller victories along the way. 'Then end users stick with you when the inevitable bumps arise,' he says (Radosevich, 1999b).

# Project Software Measurement

Project software measurement is an integral part of a successful IT project. A system consists of hardware and software. Hardware can be seen and touched. Software cannot be seen nor touched, but it is essential to the success of a computer system. The measurement process of software emphasizes software error prevention, fault detection, and removal. The measurements depend on project constraints such as resources, schedule, and performance.

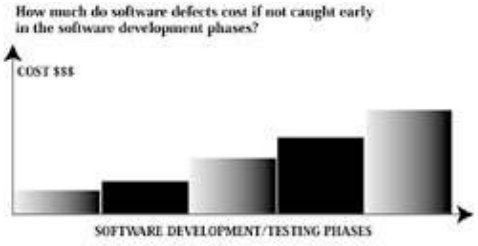
A software error is usually a programmer action or omission that results in a fault. A fault is a software defect that causes a failure, and a failure is the unacceptable departure of a program operation from program requirements.

A difference exists between hardware failure rate (Figure 7−5, *A*) and software failure rate. When the hardware component is first manufactured, the initial number of faults is high but then decreases as the faulty components are identified and removed or the components stabilize. The component then enters the useful life phase, where few if any faults are found. As the component physically wears out, the fault rate starts to increase.




7−5 A: Bathtub curve (hardware defect);

Software, however, has a different fault or error identification rate. The error rate is at the highest level at integration and test. As the software is tested, errors are identified and removed (Figure 7−5, *B*). This removal continues at a slower rate during its operational use, the number of errors continually decreasing assuming that no new errors are introduced. Software does not have moving parts and does not physically wear out as does hardware, but it does outlive its usefulness and become obsolete.




7−5 B: Software defect

The focus concerning software errors must be on comprehensive requirements and a comprehensive testing plan. The IT project manager must do the following:

- Start with the requirements, ensuring that the product developed is the one specified and that all requirements clearly and accurately specify the final product functionality.
- Ensure that the code can easily support sustaining engineering without infusing additional errors.
- A comprehensive test program is included that verifies all functionality stated in the requirements.

# Tailoring Project Measurement

Tailoring project measurement is a process (Figure 7−6). The objective of the measurement tailoring process is to define the measures that provide the greatest insight into project issues at the lowest cost. Issues are real or potential obstacles to the achievement of project objectives. The tailoring process is composed of the following three activities:

1. ***Identification and prioritization of project−specific issues.*** The IT project manager derives issues from project−context information, management experience, and risk−assessment results. The manager assigns priorities to each issue to establish its relative importance as a factor in selecting appropriate measures.
2. ***Selection of appropriate measures to address the project−specific issues.*** The selection activity employs a defined framework that maps common project issues to measurement categories to measures. These selections result in measurement data requirements that can be incorporated into the project book.
3. ***Integration of the measures into the developer's software process.*** The project manager must consider the suitability of the selected measures in the context of the developer's software process and overall technical approach. The manager should use measurement requirements not to change the developer's software process but to gain insight into it. When implementing measurement on an existing project, the manager should give special consideration to existing data sources and ongoing measurement activities. The manager documents the result of the tailoring process in a project measurement plan.

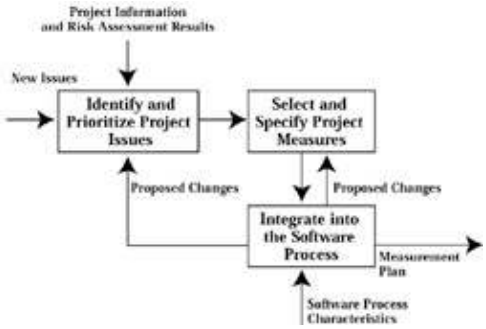



Figure 7−6: Tailoring process

# Practical Software Measurement

Practical software measurement (PSM) provides the IT project manager with the objective information needed to successfully meet cost, schedule, and technical objectives on software−intensive projects. The project issues drive PSM. The PSM describes software measurement as a systematic but flexible process that is an integral part of the overall project management structure. PSM is based on actual software measurement experience on various projects. PSM treats measurement as a flexible process, not a predefined list of graphs or reports. The process is adapted to address the specific software issues, objectives, and information requirements unique to each project. The PSM measurement process includes the following set of industry best practices (Roedler, 1999):

- ***Schedule.*** The schedule measurement includes work unit progress and milestone performance. The work unit progress covers the status of the problem report, management tracking, requirements, various phases of the life cycle, technical reviews completed, and change requests. The milestone performance covers dates, schedule dependencies, lead time, slack time, and the critical path method

(CPM).

- ***Cost.*** The cost measurement includes personnel, financial, and resource measurements. The personnel measurement covers effort and staffing. The financial measurement covers cost and earned value (EV). The resource measurement covers quantity, availability, and use.
- ***Product.*** The product measurement covers functional and product size and stability. Functional size and stability contain requirements; system functions; requirements added, deleted, and changed; and requirement traceability forward and backward. Product size and stability contain interfaces and database size.
- ***Quality.*** The quality measurement includes functional correctness, efficiency, reliability, usability, maintainability, and portability. Functional correctness covers problem reports and defects. Efficiency covers throughput, use, and time. Reliability covers system failures. Usability covers learning difficulty, whether the system is user hostile or user friendly, operational errors, and customization difficulty. Maintainability covers maintenance time and update feasibility.
- ***Performance.*** The performance measurement includes evaluation of efficiency, effectiveness, and updates. The evaluation process covers capability and audit. The efficiency evaluation process covers productivity and cycle time. The effectiveness process covers completion of system development and maintenance phases. The update process covers the size and effort of rework.
- ***Technology influence.*** The technology influence measurement includes maturity and effectiveness of technology. The technology maturity includes the stability of technology and its adequacy for the IT system. The technology influence covers its implementation and functionality for the specific system.
- ***Customer.*** The customer measurement covers satisfaction with the system and feedback. The customer feedback includes award fee amounts, survey results, and number of commendations and complaints.

PSM integrates the measurement requirements into the software developer's process. The project manager tailors the measurement set for each project to ensure that the measurement process is cost−effective and that measures provide meaningful and usable results.

PSM defines an issue−driven analysis approach, which helps the project manager make informed software decisions. The PSM analysis approach incorporates the use of multiple measures and nonquantitative program data to identify and evaluate software problems.

PSM defines a nonprescriptive measurement and provides a mechanism for the objective communications necessary within an integrated product team (IPT).

PSM provides a basis for enterprise−level software management. PSM is designed to help the manager put measurement into practice at the project level, thereby providing the data required to address enterprise−level software performance, process improvement, and business−related questions. PSM also supports IT performance measurement requirements.

# Project Measurement Checklist

- Ensure that everyone in the project understands the capabilities and limitations of the measurement process.
- Start small. Implement only a few measures to address key issues, and show how the measurement results support both the project and higher−level management objectives.
- Do not allow anyone in the project to use measurement to evaluate individual or work−group performance.
- Ensure that only the required measures are implemented based on the issues and objectives of the project. Do not collect data that are not needed. The measurement process must be cost−effective to

succeed.

- Make the measurement data and information available to everyone on the project.
- Initially implement the measurement process with basic, commercially available database, spreadsheet, word processing, and presentation graphics applications.
- Make measurement an integral part of the project.
- Ensure that all users at all levels understand what the measurement data represent. This understanding is vital to the proper interpretation of the measurement analysis results.