Fundamentals of Measurements

Educational Objective: To review the fundamentals of software measurement, to illustrate that measurement plays a central role in process management, and to discuss questions that lead to measurement categorization and selection.

- Definition
- Measurements and Process Management
- Measurement Principles
- Measurement Questions and Measurement Selection
- Measurement Process

Metrics and Measurements

A phenomenon will be said to be controlled when, through the use of past experience, we can predict, at least within limits, how the phenomenon may be expected to vary in the future.

Walter A. Shewharts, 1931

You can't control what you can't measure. Measurement costs money . . . if you think that cost is high, consider the cost of being out of control.

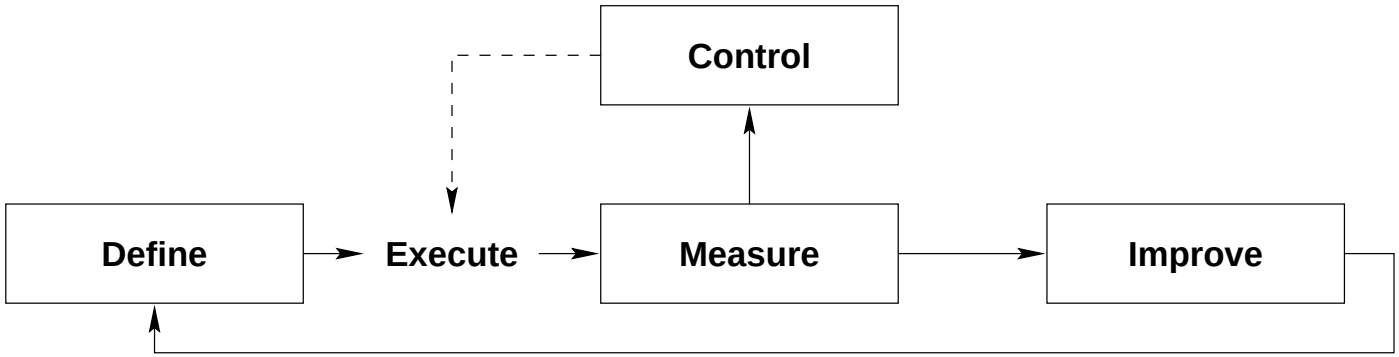
Tom DeMarco

Measurements: Definition

Measurement is the quantification of attributes of an object or a process; its objectives are:

- Collect measurements to control processes
- Controlled processes lead to stable processes
- Stable processes enable managers to predict their behavior and results

Measurements for Process Management



Define the Process. Objective: to create a disciplined process environment

Measure the Process. Objective: to collect data that measure the performance of each process and detect deviations from acceptable performance (and to retain such data to predict future performances)

Control the Process: Objective: to keep a process within its normal performance expectations (boundaries)

Improve the Process. Objective: to make changes that improve the existing capabilities of a process (e.g., wrt efficiency)

Why Measurements?

Adopted from the *Practical Software Measurement* (Version 3.1a, 1998):

- Serve as communication tool: measurement allows software issues to be explicitly identified, prioritized, tracked and communicated at all levels
- Allow pro-active problem resolution instead of waiting for something bad to happen
- Provide for project tracking: measurements accurately describe the status of project
- Allow decision making: measures help manager make, defend and justify important decisions

Software Measurement Principles

- Project objectives (e.g., schedule and cost) drive the measurement requirements; thus measures should be collected in support of predetermined objectives
- Measure processes not people
- Collect and analyze data at a level of detail sufficient to identify and isolate software problems
- Collected data must be carefully reviewed for correction, consistency, and completeness

Software Measurement Principles (continued)

- A systematic measurement analysis process should be used to provide useful data to the management for decision making
- Integrate software measurement collection and analysis into the project management process throughout the software life cycle
- Interpret the measurement results in the context of other project information (e.g., if behind in unit testing, could it be because you are not fully staffed?)
- Use the measurement as a basis for objective communication

Life-Cycle Application of Measurements

Major phases of a software life-cycle:

- Planning
- Development
- Maintenance

Project Planning: a development plan is produced that contains estimation of cost, resources, duration

Past performance measurement can be used to evaluate the capability of a potential developer (or your own internal teams) and more accurately estimate cost, efforts, duration

Life-Cycle Application of Measurements (continued)

Development:

- Requirements analysis: measure stability, schedule, progress
- Design: measure progress, complexity, technical accuracy
- Coding: measure progress, product quality, complexity, size
- Integration and testing: measure progress, product quality

Maintenance: measure defect and correction rates, costs

Measurement: Who Does What?

- **Executive manager:** use measures to make key enterprise decisions
- **Project manager:** reviews measurement analysis results and acts upon measurement information
- **Measurement analyst:** collect and analyze measurements and report the results throughout the project organization
- **Project development teams:** collect measurement data on a periodic basis; uses measurement results in software engineering efforts; provides data to the analysts

Measurement Questions

To be cost effective, a measurement program must be designed and targeted to support the business goals and provide reliable information for decision making

Typical initial questions

- What should be measured?
- How should the attributes be identified and captured?
- When and how is the measurement data analyzed?
- What is a measurement analysis telling us?
- What decisions will be made?

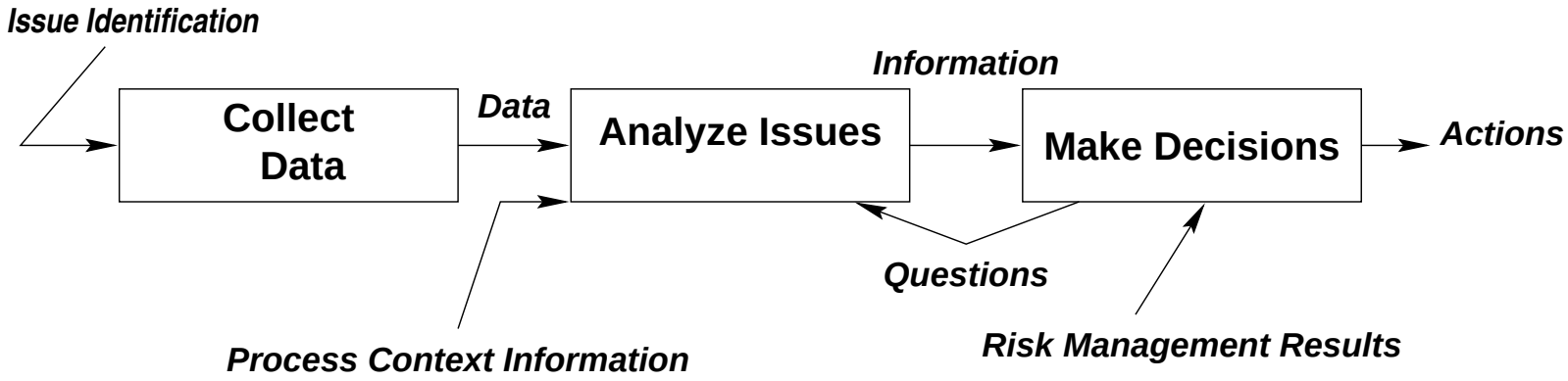
Selection of Appropriate Measures

The following table shows measures for certain issues and categories:

Issue	Category	Measures
Schedule and progress	*Milestone performance*	Milestone dates
	Work unit progress	Test case status, Paths tested Reviews completed Requirement status
Resources and cost	*Personnel*	Effort, Experience
	Financial performance	Cost, Earned value
	Environment availability	Resource availability dates Resource utilization
Growth and stability	*Product size*	LOC, Elements
	Functional size	Requirements
	Stability	Incident report status Change requests
Product quality	*Defects*	Incident reports, Defect density
	Complexity	Cyclomatic complexity
	Rework	Rework efforts, Rework size
Development performance	*Process maturity*	Maturity level
	Productivity	Product size/effort ratio

PSM Measurement Process

- Start with issue identification and prioritization
- Lines of communication must be identified
- Measurements must result in decision and actions
- Measurement process by which data is collected and transformed into information:



Measurement Process: 1. Collect Data

Collect and Process Data

- Obtain data from all sources (plans, reports)
  - Develop clear and concise definitions for data collection
  - Collect data at a level appropriate to localize problem
  - Expect the data to be noisy
- Define a procedure
  - Identify responsible individuals
  - Define when, where and how data is collected
  - Define the procedure for data collection
  - Use standard templates for collecting and documenting collected data

Measurement Process: 1. Collect Data

- Verify data
  - Correct type?
  - Correct format?
  - Correct range?
  - Complete?
  - Question unusual trends in data
  - Normalize inconsistent data
- Normalize data: A process of converting raw data into a different unit of measure (e.g., efforts in hours to effort in months) so it can be compared or combined with other data
- Document for future (retain): establish appropriate databases to save measurement data for future uses

Measurement Process: 2. Analyze Issues

Analyze issues:

- Trust the data? Is there really a problem? How big is it?
- What is the problem scope? What is causing it? What are alternatives?
- Feasibility analysis and performance analysis (actual adherence to plans and estimate, e.g., available personnel effort conforms to the plan?)

Define and generate indicators:

- An ***Indicator***: A measure or combination of measures that provides insight into a software issue; usually compares planned (baseline) vs. actual values; often displayed graphically
- Baselines are needed to make meaningful analysis and may be derived from plans, rules of thumb, norms, totals and averages, typical values
- Indicators should be generated regularly

Measurement Process: 2. Analyze Issues — Types of Analysis

Estimation: Early in the process, the focus is on estimation to support project planning — typical issues are related to completeness and clarity of definitions for software size, effort, and schedule

Feasibility Analysis: As plans near completion, the focus shifts to typical issues related to whether plans and targets are realistic and achievable

Performance Analysis: Once the project has begun, performance analysis becomes major concern — typical issues are related to whether development efforts are meeting defined plans and key milestones

Measurement Process: 3. Make Decisions

- The results must be clearly understood by decision-makers
- Alternative courses of action should be considered; action must be taken to realize any benefit from measurement
- Desired action may not always be possible; may have to optimize within project constraints

Practical Software Measurement (PSM)

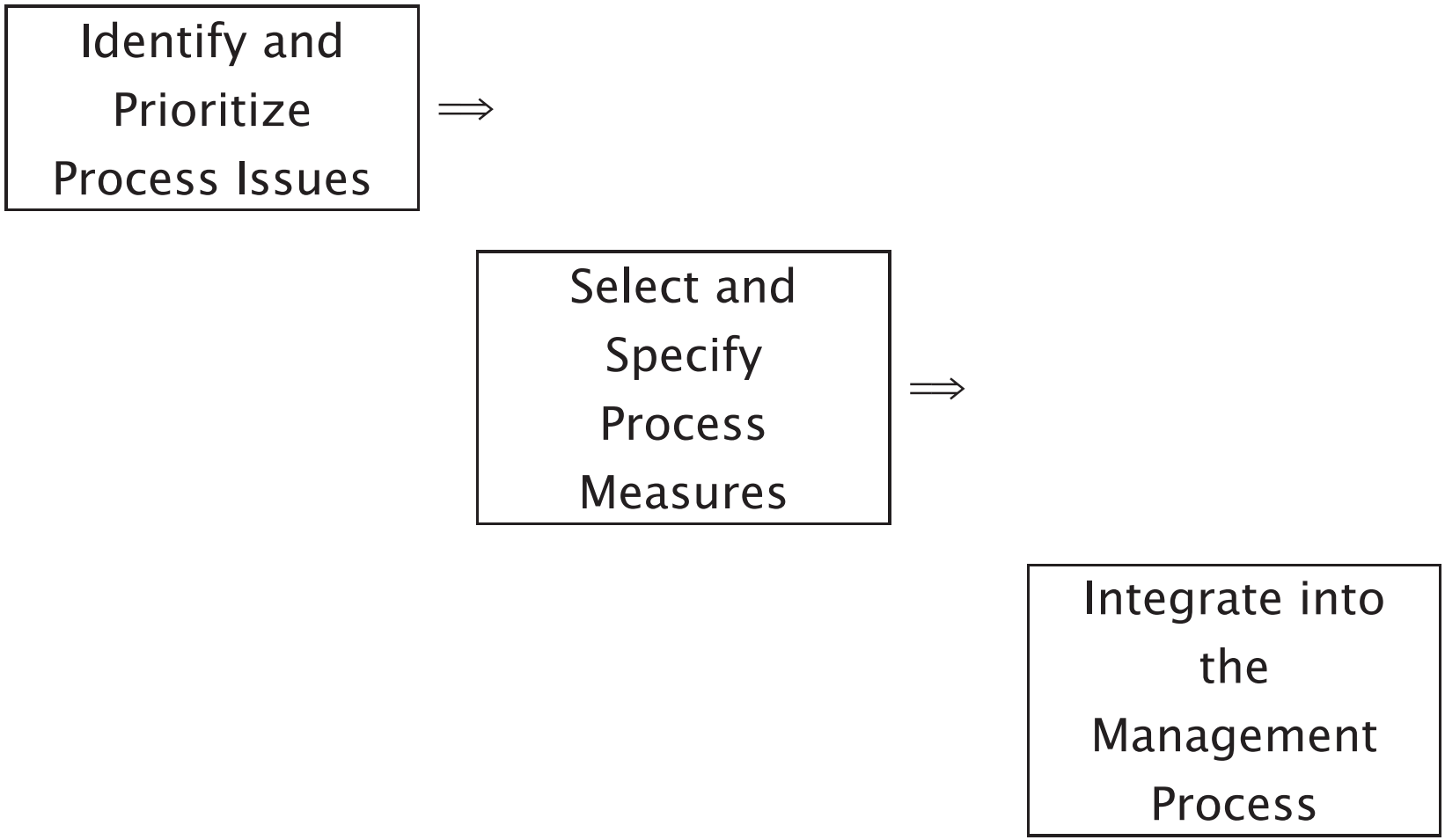
Educational Objectives: To introduce the Practical Software Measurement

- PSM View of Measurement
- PSM Measurement Process
  - Data Collection
  - Issue Analysis
  - Decision Making
- PSM Tailoring Mechanism
- PSM Measurement Categories

A Look at Practical Software Measurements (PSM)

- Office of the Under Secretary of Defense for Acquisition and Technology: developed collaboratively by many individuals and organizations
- Objectives: Assist project and technical managers meet software cost, schedule and technical objectives
  - Help project and technical managers anticipate what can go wrong
- Provides a basis for objective communication and informed decision making
- Provides a tailoring mechanism that is adaptive to specific development issues and objectives
- Defines several measurement categories

PSM: Tailoring Software Measures

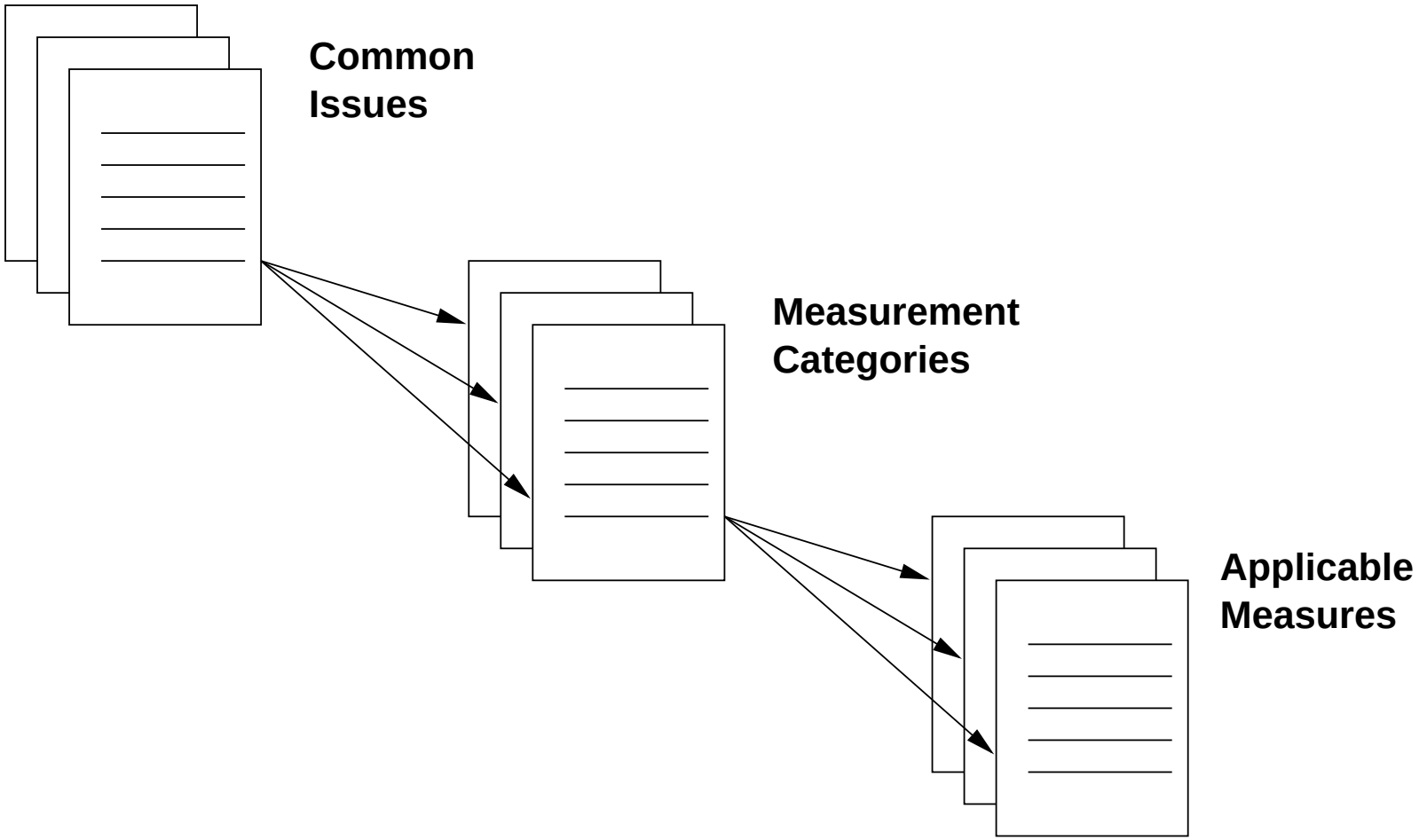


Selection of Appropriate Measures

The following table shows measures for certain issues and categories:

Common Issue	Category	Measures
Schedule and progress	*Milestone performance*	Milestone dates
	Work unit progress	Test case status, Paths tested Reviews completed Requirement status
Resources and cost	*Personnel*	Effort, Experience
	Financial performance	Cost, Earned value
	Environment availability	Resource availability dates Resource utilization
Growth and Stability	*Product size*	LOC, Elements
	Functional size	Requirements
	Stability	Incident report status Change requests
Product quality	*Defects*	Incident reports, Defect density
	Complexity	Cyclomatic complexity
	Rework	Rework efforts, Rework size
Development Performance	*Process maturity*	Maturity level
	Productivity	Product size/effort ratio

PSM: Measurement Tailoring Mechanisms



PSM: Common Software Issues

Software issue: An area of concern that could impact a project's objective

PSM defines a simple mapping from the common software issues to related measurement categories, and then to individual measures in each category

Common issues:

1. Schedule and progress
2. Resources and cost
3. System growth and stability
4. Product quality
5. Development performance

PSM: Common Issues as Measurement Categories

1. Schedule and Progress

- ***Milestone Performance*** measures provide basic schedule and progress information for key software development activities and events; they also identify and assess dependencies among software development activities and events
- ***Work Unit Progress*** measures address progress based on the completion of work units that combine incrementally to form a completed product; they are every effective for assessing progress at any point in the project

PSM: Common Issues as Measurement Categories

2. Resources and Cost

- ***Personnel*** measures identify the amount of effort expended on defined software activities; they also measure and characterize the number of and experience of personnel assigned to a project
- ***Cost Performance*** measures report the difference between budgeted and actual costs for a specific product or activity; they are used to assess whether the project can be completed within cost and schedule constraints and to identify potential cost overruns
- ***Environment Availability*** measures address the availability and utilization of tools and system resources

PSM: Common Issues as Measurement Categories

3. Growth and Stability

- ***Product Size*** measures quantify the physical size of a software product or the growth due to incremental change
- ***Functional Size*** measures quantify the functionality of a software product or the growth due to incremental change
- ***Stability*** measures provide information about the amount and frequency of change to software functionality

PSM: Common Issues as Measurement Categories

4. Product Quality

- ***Defect*** measures identify the number of problem reports, defects, or failures in the software products and/or processes (e.g., incident reports) and are used to evaluate the quality of the products and development process
- ***Standards Compliance*** measures define the extent to which the software conforms to established coding standards
- ***Complexity*** measures quantify the structure of software components, based on the number of and intricacy of interfaces and branches, the degree of nesting, the types of data structures, and other system characteristics; they provide indicators of the need to redesign and the relative amount of testing required for a software component (e.g., cyclomatic complexity)
- ***Rework*** measures address the amount of rework effort due to defects in completed work products (e.g., documents, design, code)and are used to evaluate the efficiency of the development organization

PSM: Common Issues as Measurement Categories

5. Development Performance

- ***Process Maturity*** measures address the capability of the software development processes within an organization and may be used to predict the ability of an organization to best address the issues and constraints of a development project
- ***Productivity*** measures identify the amount of software product produced per unit of effort; they are widely used as an indication of whether a project has adequate funding and schedule relative to the amount of software to be developed

PSM Measures

PSM defines more than 40 different (though some may be related) measures; less than half will be discussed

For each measure, the PSM category and PSM issue will be identified; each description will be in terms of:

- **Selection Guidelines** (e.g., project type application and when to apply)
- **Specification Guidelines** (e.g., data and implementation requirements)

Review Common Issues

Common Issue:

Schedule and Progress

Measurement Category: Milestone Performance

Milestone Performance measures provide basic schedule and progress information for key software development activities and events; they also identify and assess dependencies among software development activities and events

Project Application: Applicable to most projects, applicable to all software process models; useful during planning, development and maintenance

Measures in this Category: Milestone dates

Measure: Milestone Dates

Category:	Milestone Performance
Common Issue:	Schedule and Progress

This measure answers questions such as: is the current schedule realistic? How much progress has been made? Has the schedule changed? What is the completion date?

Measure: Milestone Dates

Selection Guidelines

- Applicable to all domains
- Required data is usually easily obtained
- More detailed milestones provide a better indication of progress and allow earlier identification of problems
- Usually applied during requirements definition, design, implementation and testing

Specification Guidelines

- Typical data items: Start date, End date, Due date
- Typical attribute: Status
- Typically collected for each element or project

Measurement Category: Work Unit Progress

Work Unit Progress measures address progress based on the completion of work units that combine incrementally to form a completed product; they are very effective for assessing progress at any point in the project

Project Application: Applicable to most projects, applicable to all software process models; useful during planning, development and maintenance

Measures in this Category: Requirements status, Design status, Element coding status, Test case status, Test coverage, Reviews completed, Change request status

Additional Information: Elementss may be defined differently for

each project and can be modules, copybooks, objects, interfaces, screens, reports, commercial off-the-shelf (COTS) products, etc.

Measure: Requirements Status

Category:	Work Unit Progress
Common Issue:	Schedule and Progress

This measure answers questions such as: are the requirements being defined as scheduled? Is implementation of the requirements behind schedule? Are dependencies being met?

Measure: Requirements Status

Selection Guidelines

- Applicable to all domains; useful during development and maintenance
- Requires disciplined requirements traceability
- Usually applied during requirements definition, design, implementation and testing

Measure: Requirements Status

Specification Guidelines

- Typical data items: Number of requirements; number of requirements traced to the detailed specification, software components and test specifications; number of requirements tested successfully
- Typical attributes: Quantity, Status
- Typically collected for each requirements specification

Measure: Test Case Status

Category:	Work Unit Progress
Common Issue:	Schedule and Progress

This measure answers questions such as: is test progress sufficient to meet the schedule? Is the planned rate of testing realistic? What functions are behind schedule?

Measure: Test Case Status

Selection Guidelines

- Applicable to all domains; normally used during development and maintenance
- Need disciplined test planning and tracking
- Can be applied during each test sequence such as integration and regression test
- Relatively easy to collect; most projects define and allocate a quantifiable number of test case to each test sequence
- Usually applied during component test, project test. integrated system test

Measure: Test Case Status

Specification Guidelines

- Typical data items: Number of test cases, number of test cases attempted, number of test cases passed
- Typically collected for each activity test

Measure: Test Coverage (Paths Tested)

Category:	Work Unit and Progress
Common Issue:	Schedule and Progress

This measure answers questions such as: What percentage of the new and changed code has been tested? What percentage of the legacy code has been regression tested?

Measure: Test Coverage (Paths Tested)

Selection Guidelines

- Applicable to all domains and most types of projects (those with high reliability requirements)
- Useful during development and maintenance; not generally used for COTS or reused code
- Usually applied on a cumulative basis across all tests to determine coverage
- Difficult to collect; requires automated tools
- Usually applied during unit test and component test

Measure: Test Coverage (Paths Tested)

Specification Guidelines

- Typical data items: Amount of new and changed code, New and changed code tested, Number of tested successfully
- Typically collected for each unit or component

Measure: Reviews Completed

Category:	Work Unit Progress
Common Issue:	Schedule and Progress
	(e.g., Process Compliance)

This measure answers questions such as: Are development review activities progressing as scheduled? Do the completed products meet the defined standards? What components have failed their reviews? Is the process being followed?

Measure: Reviews Completed

Selection Guidelines

- Applicable to all domains; normally used during development and maintenance; not normally used on projects integrating COTS and reusable components
- Easy to collect if formal reviews are part of the development process
- Usually applied during requirements definition, design, implementation, test development, and integration

Measure: Reviews Completed

Specification Guidelines

- Typical data items: Number of reviews, number of reviews scheduled, number of reviews completed
- Typical attribute: Quantity, Ratio of quantity to size
- Typically collected for each element or project

Review Common Issues

Common Issue:

Personnel and Cost

Measurement Category: Personnel

Personnel measures identify the amount of effort expended on defined software activities; they also measure and characterize the number of and experience of personnel assigned to a project

Project Application: Applicable to most projects, applicable to all software process models; useful during planning, development and maintenance

Measures in this Category: Effort, Staff experience

Additional Information: Software activities may include any activity during the planning, development and maintenance

Measure: Effort

Category:	Personnel
Common Issue:	Resource and Cost

This measure answers questions such as: Are development resources being applied according to plan? Are certain tasks or activities taking more/less effort than expected?

Measure: Effort

Selection Guidelines

- Applicable to all domains; useful during planning, development, and maintenance
- Data should be available from most projects; usually driven from financial accounting and time reporting system
- All labor hours applied to the software tasks should be collected including overtime; if hours are not explicitly provided, data may be approximated from staffing or cost data
- Usually applied during requirements analysis, design, implementation, testing, and maintenance

Measure: Effort

Specification Guidelines

- Typical data item: Number of labor hours
- Typical attributes: Organization, labor category
- Typically collected for each software activity

Measurement Category: Financial Performance

Cost Performance measures report the difference between budgeted and actual costs for a specific product or activity; they are used to assess whether the project can be completed within cost and schedule constraints and to identify potential cost overruns

Project Application: Applicable to most projects, applicable to all software process models; useful during planning, development and maintenance

Measures in this Category: Earned value, Cost, Resources expended, Estimate to complete

Measure: Cost

Category:	Cost Performance
Common Issue:	Resource and Cost
	(e.g., Budget and Resource Commitment)

This measure answers questions such as: Are project costs in accordance with estimates? Will the target budget be achieved or will there be an overrun or surplus?

Measure: Cost

Selection Guidelines

- Applicable to all domains and projects of all sizes and types
- Useful during project planning, development and maintenance phases
- Data should come from an automated accounting systems and should be relatively easy to collect at a high level
- Usually applied during requirements analysis, design, implementation and testing

Measure: Cost

Specification Guidelines

- Typical data item: Cost (in dollars), Resources (in hours)
- Typical attribute: Organization, Resource type
- Typically collected for each software activity

Measurement Category: Environment Availability

Environment Availability measures the availability and utilization of tools and facility resources. Resources include those used for development, maintenance, and operation. These measures are used to address the adequacy of resources.

Project Application: Applicable to all projects, applicable to all software process models; useful during planning, development and maintenance

Measures in this Category: Resource availability dates, Resource Utilization

Measure: Resource Availability

Dates

Category:	Environment Availability
Common Issue:	Resource and Cost
	(e.g., Process Plans and Cost)

This measure answers questions such as: Are key resources available when needed? is the availability of support resources impacting progress?

Measure: Resource Availability Dates

Selection Guidelines

- Applicable to all domains; more important for projects with constrained resources; Useful during development and maintenance phases
- Data is generally easily obtained from documentation
- Resources may include software, hardware, integration, and test facilities, tools or office space
- Normally only key resources are tracked
- Usually applied during requirements analysis, design, implementation and testing

Measure: Resource Availability Dates

Specification Guidelines

- Typical data items: availability date
- Typically collected for key resource

Measure: Resource Utilization

Category:	Environment Availability
Common Issue:	Resource and Cost
	(e.g., Workload Commitments)

This measure answers questions such as: Are sufficient resources available? How efficiently are resources being used?

Measure: Resource Utilization

Selection Guidelines

- Applicable to all domains; more important for projects with constrained resources
- Relatively easy to collect at a high level
- Resources may include software, hardware, integration, and test facilities, tools or office space, but often is focused on personnel
- Normally constrained resources are tracked
- Usually applied during requirements analysis, design, implementation and testing

Measure: Resource Utilization

Specification Guidelines

- Typical data items: Resource hours, allocated hours, scheduled hours, available hours, remaining hours
- Typically collected for each key resource

Review Common Issues

Common Issue:

Growth and Stability

Measurement Category: Product Size

Product Size measures quantify the physical size of a software product or growth due to incremental change

Project Application: Applicable to most projects, applicable to all software process models; useful during planning, development and maintenance

Measures in this Category: Lines of code, Elements, Word of Memory, Database size

Additional Information: Elements may be defined differently for each project and can be modules, copybooks, objects, interfaces, screens, reports, COTS, etc.

Measure: Lines of Code

Category:	Product Size
Common Issue:	Growth and Stability

This measure answers questions such as: How accurate were the size measures on which the schedule and effort plans were based? How much has the software size changed? In what component have changes occurred? Has the size allocated to each incremental build changed?

Measure: Lines of Code

Selection Guidelines

- Applicable to all domains; used for projects of all sizes
- Most effective for traditional high level languages such as COBOL
- Not usually tracked for COTS (unless changes are made)
- Useful during planning, development, and maintenance phases
- Defines lines of code (LOC); LOC from different languages are not equivalent
- New and modified lines count at 100%
- Usually difficult to generate accurate estimates early in the

project

- Actuals can be measured with tools
- Usually applied during requirements definition, design, implementation, and in production

Specification Guidelines

- Typical data items: number of LOC, number of LOC added, number of LOC modified (sometimes number of LOC deleted)
- Typical attributes: version, source (new, COTS), language, delivery status
- Typically collected for each element
- LOC definition may include logical lines, physical lines, comments, executables, data declarations, and compiler directives

Measure: Elements (Components)

Category:	Product Size
Common Issue:	Growth and Stability

This measure answers questions such as: How many components need to be implemented and tested? How much has the approved software baseline changed? Have the components allocated to each incremental build changed?

Measure: Elements (Components)

Selection Guidelines

- Applicable to all application domain and projects of all sizes
- Usually not tracked for COTS software (unless changes are made)
- Useful during planning, development, and maintenance phases
- Requires a well-defined and consistent component allocation structure
- Required data is generally easy to obtain (from software design tools)
- Deleted and added components are relatively easy to collect;

modified components are often not tracked

- Usually applied during requirements analysis, design, implementation, integration,and test

Specification Guidelines

- Typical data items: Number of units, number of units added, number of units deleted, number of units modified
- Typical attributes: Version, source, delivery status
- Typically collected for each element or CI

Measurement Category: Functional Size

Functional Size measures quantify the functionality of a software product or the growth due to incremental change

Project Application: Applicable to most projects, applicable to all software process models; useful during planning, development and maintenance

Measures in this Category: Requirements, Function points

Additional Information: Function point data collection requires a defined method or tool and is often labor intensive

Measure: Requirements

Category:	Functional Size
Common Issue:	Growth and Stability

This measure answers questions such as: Have the requirements allocated to each incremental build changed? Are requirements being deferred to later builds? How much has software functionality changed? What components have been affected the most?

Measure: Requirements

Selection Guidelines

- Applicable to all domains; applicable to projects that track requirements
- Useful during planning, development, and maintenance phases
- Requires a good requirements traceability process
- Count changes against a baseline that is under formal configuration control
- Usually applied during requirements analysis, design, implementation, integration,and test

Measure: Requirements

Specification Guidelines

- Typical data items: Number of requirements, Number of requirements added, deleted, modified
- Typically collected for each requirement specification

Measurement Category: Stability

Stability measures quantify the amount of change occurring during the development life cycle

Project Application: Applicable to most projects, applicable to all software process models; useful during planning, development and maintenance

Measures in this Category: Requirements change requests, Design addenda, Rework items, Incident reports status

Measure: Incident Report Status

Category:	Stability
Common Issue:	Schedule and Progress
	(e.g., Readiness for Production)

This measure answers questions such as: Are known problem reports being closed at a sufficient rate to meet the test completion date? Is the product maturing? When will testing be complete?

Measure: Incident Report Status

Selection Guidelines

- Applicable to all domains and types of projects; useful during development and maintenance
- Data is generally available during integration and test;
- Usually applied during implementation and testing

Specification Guidelines

- Typical data items: Number of software problems reported, number of software problems resolved
- Typical attributes: Priority, valid/invalid
- Typically collected for each project or requirement

Measure: Change Requests

Category:	Stability
Common Issue:	Growth and Stability

This measure answers questions such as: how many change requests have been written? is the backlog of open change requests declining? is the rate of new change requests increasing or decreasing?

Measure: Change Requests

Selection Guidelines

- Applicable to all domains and all sizes of project
- Useful during development phase
- Data should be available for most projects
- Usually applied during requirements analysis, design, implementation, integration,and test

Measure: Change Requests

Specification Guidelines

- Typical data items: Number of software change requests written, Number of software change requests open, Number of software change requests assigned to a version, Number of software change requests resolved
- Typical attributes: Version, priority, approved/unapproved, valid/invalid, change classification (defect, enhancement)
- Typically collected for each requirement specification and design specification

Measurement Category: Defects

Defect measures identify the number of problem reports, defects, and failures in the software products and/or processes

Project Application: Applicable to most projects, applicable to all software process models; useful during planning, development and maintenance

Measures in this Category: Rework items, Incident reports, defect density, Deviations from standards, System availability

Additional Information: A defect is a product's non-conformance with its specification; an incident report is a documented description of a defect; defect measures are extremely useful during all phases of a product life cycle

Measure: Incident Reports

Category:	Defects
Common Issue:	Product Quality

This measure answers questions such as: How many critical problem reports have been written? How many problem reports are open? What are their priorities?

Measure: Incident Reports

Selection Guidelines

- Applicable to all domains and all sizes and types of projects
- Useful during planning, development, and maintenance phases
- Requires a disciplined problem reporting process
- Usually applied during implementation and testing, but also used during production maintenance

Specification Guidelines

- Typical data items: number of problem reports, average age of problem reports
- Typical attributes: version, priority, problem report status code
- Typically collected for each CI (or equivalent)

Measure: Defect Density

Category:	Defects
Common Issue:	Product Quality

This measure answers questions such as: What is the quality of the software? What components have a disproportionate amount of defects? What components require additional testing or review? What components are candidates for rework?

Measure: Defect Density

Selection Guidelines

- Applicable to all domains and all sizes and types of projects
- Useful during planning, development, and maintenance phases
- Requires a disciplined problem reporting process and a method of measuring software size
- Actuals are relatively easy to collect
- Usually applied during requirements analysis, design, implementation, integration,and test

Specification Guidelines

- Typical data items: number of defects, number of LOC
- Typical attributes: version, priority, source (new, COTS), language

Measurement Category: Complexity

Complexity measures quantify the structure of software components, based on the number of and intricacy of interfaces and branches, the degree of nesting, the types of data structures, and other system characteristics; they provide indicators of the need to redesign and the relative amount of testing required of any component

Project Application: Applicable to most projects, applicable to all software process models; useful during planning, development and maintenance

Measures in this Category: Cyclomatic complexity

Additional Information: Reducing complexity requires rework to redesign or recode

Measure: Cyclomatic Complexity

Category:	Complexity
Common Issue:	Product Quality

This measure answers questions such as: How many complex components exist in this project? What components are the most complex? What components should be subject to additional testing? What is the minimum number of reviews and test cases required to test the logical paths through a component?

Measure: Cyclomatic Complexity

Selection Guidelines

- Applicable to all domains
- Applicable to projects with testability, reliability, or maintainability concerns
- Not generally used with COTS or reused code
- Useful during development and maintenance phases
- Cyclomatic complexity does not differentiate between type of control flow
- Cyclomatic complexity does not address data structures
- Relatively easy to obtain when automated tools are available such as C and Ada

- Estimates are not generally derived, but a desired threshold or expected distribution may be specified

Specification Guidelines

- Typical data items: cyclomatic complexity rating
- Typical attribute: version
- Typically collected for each unit or equivalent

Review Common Issues

Common Issue:

Development Performance

Measurement Category: Rework

Rework measures address the amount of rework due to defects in completed work products (e.g., documents, design, code) and are used to evaluate the quality of the products and development process; they provide information on how much software must be recoded and how much effort is required for corrections

Project Application: Applicable to most projects, applicable to all software process models; useful during planning, development and maintenance

Measures in this Category: Rework size, Rework effort

Additional Information: Data collection is difficult and often labor intensive; requires a consistent process for effort allocation to rework/non-rework categories

Measure: Rework Effort

Category:	Rework
Common Issue:	Development Performance

This measure answers questions such as: How much effort was expended on fixing defects in a product? What software activity required the most rework? Is the amount of rework impacting cost and schedule?

Measure: Rework Effort

Selection Guidelines

- Applicable to all domains
- Applicable to most development processes; in a rapid prototype process, it is only applicable to the final version of the software product
- useful during development and maintenance phases
- Difficult to collect; rework effort should only include effort associated with correcting defects (not adding enhancements)
- Rework cost and schedule estimate should be included in the development plan

Measure: Rework Effort

Specification Guidelines

- Typical data item: labor hours due to rework
- Typical attributes: organization, labor category, version
- Typically collected for each software activity

Measurement Category: Process Maturity

Process Maturity measures address the capability of the software development processes within an organization and may be used to predict the ability of an organization to best address the issues and constraints of a development project

Project Application: Applicable to most projects, applicable to all software process models; useful during planning, development and maintenance

Measures in this Category: CMM level

Additional Information: Measure obtained through formal assessment for certification; there is subjectivity in determination of the process maturity

Measure: Capability Maturity Model (CMM) Level

Category:	Process Maturity
Common Issue:	Development Performance

The CMM level measure reports the rating (1-5) of a software development organization's software development process, as defined by the SEI. The measure is the result of a formal assessment of the organization's project management and software engineering capabilities.

Measure: Capability Maturity Model (CMM) Level

Selection Guidelines

- Applicable to all domains; normally measured at the organizational level
- Useful during project planning, development, and maintenance

Specification Guidelines

- Typical data items: CMM rating
- Typical attributes:none
- Typically collected for at the organization level

Measurement Category: Productivity

Productivity measures identify the amount of software product produced per unit of effort; they are widely used as an indication of whether a project has adequate funding and schedule relative to the amount of software to be developed

Project Application: Applicable to most projects, applicable to all software process models; useful during planning, development and maintenance

Measures in this Category: Product size/effort ratio; Functional size/effort ratio

Measure: Product Size/Effort Ratio

Category:	Productivity
Common Issue:	Development Performance

This measure answers questions such as: is the developer's production rate sufficient to meet the completion date? How efficient is the developer at producing the software product? Is the planned/required software productivity rate realistic?

Measure: Product Size/Effort Ratio

Selection Guidelines

- Applicable to all domains; used for projects of all size
- Not generally used for COTS or reused software
- Estimates are often used during project planning; estimates and actual used during development
- The environment, language, tools, and personnel experience will affect productivity achieved
- Definitions should specify those elements of effort that are included, such as project management and documentation
- Usually applied during requirements analysis, design, implementation, integration,and test

Measure: Product Size/Effort Ratio

Specification Guidelines

- Typical data items: number of LOC, number of labor hours
- Typical attributes: version, language
- Typically collected for each project

Conclusions

To summarize and conclude

- Measurement is the foundation for effective software management
- Measurement results are required to objectively manage risk and evaluate performance
- PSM provides a basis for meeting measurement requirements