Project Management Concepts

Management Spectrum

People — the most important element of a successful project

Product — the software to be built

Process — the set of framework activities and software engineering tasks to get the job done

Project — all work required to make the product a reality

The Stakeholders

Senior managers who define the business issues that often have significant influence on the project.

Project (technical) managers who must plan, motivate, organize, and control the practitioners who do software work.

Practitioners who deliver the technical skills that are necessary to engineer a product or application.

Customers who specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome.

End-users who interact with the software once it is released for production use.

Software Team

How to lead?

How to organize?

How to collaborate?

How to motivate?

How to create good ideas?

Team Leader

The MOI Model

Motivation. The ability to encourage (by "push or pull") technical people to produce to their best ability.

Organization. The ability to mold existing processes (or invent new ones) that will enable the initial concept to be translated into a final product.

Ideas or innovation. The ability to encourage people to create and feel creative even when they must work within bounds established for a particular software product or application.

Software Team

- ***The following factors must be considered when selecting a software project team structure ...***

- the difficulty of the problem to be solved
- the size of the resultant program(s) in lines of code or function points
- the time that the team will stay together (team lifetime)
- the degree to which the problem can be modularized
- the required quality and reliability of the system to be built
- the rigidity of the delivery date
- the degree of sociability (communication) required for the project

Organizational Paradigms

closed paradigm—structures a team along a traditional hierarchy of authority

random paradigm—structures a team loosely and depends on individual initiative of the team members

open paradigm—attempts to structure a team in a manner that achieves some of the controls associated with the closed paradigm but also much of the innovation that occurs when using the random paradigm

synchronous paradigm—relies on the natural compartmentalization of a problem and organizes team members to work on pieces of the problem with little active communication among themselves

Avoid Team Toxicity

- A frenzied work atmosphere in which team members waste energy and lose focus on the objectives of the work to be performed.
- High frustration caused by personal, business, or technological factors that cause friction among team members.
- "Fragmented or poorly coordinated procedures" or a poorly defined or improperly chosen process model that becomes a roadblock to accomplishment.
- Unclear definition of roles resulting in a lack of accountability and resultant finger-pointing.
- "Continuous and repeated exposure to failure" that leads to a loss of confidence and a lowering of morale.

Agile Teams

Team members must have trust in one another.

The distribution of skills must be appropriate to the problem.

Mavericks may have to be excluded from the team, if team cohesiveness is to be maintained.

Team is “self-organizing”

An adaptive team structure

Uses elements of Constantine’s random, open, and synchronous paradigms

Significant autonomy

Team Coordination and Communication

Formal, impersonal approaches include software engineering documents and work products (including source code), technical memos, project milestones, schedules, and project control tools ,change requests and related documentation, error tracking reports, and repository data.

Formal, interpersonal procedures focus on quality assurance activities applied to software engineering work products. These include status review meetings and design and code inspections.

Informal, interpersonal procedures include group meetings for information dissemination and problem solving and “collocation of requirements and development staff.”

Electronic communication encompasses electronic mail, electronic bulletin boards, and by extension, video-based conferencing systems.

Interpersonal networking includes informal discussions with team members and those outside the project who may have experience or insight that can assist team members.

The Product Scope

Context. How does the software to be built fit into a larger system, product, or business context and what constraints are imposed as a result of the context?

Information objectives. What customer-visible data objects are produced as output from the software? What data objects are required for input?

Function and performance. What function does the software perform to transform input data into output? Are any special performance characteristics to be addressed?

Software project scope must be unambiguous and understandable at the management and technical levels.

The Process

Once a process framework has been established

Consider project characteristics

Determine the degree of rigor required

Define a task set for each software engineering activity

Task set = Software engineering tasks + Work products + Quality assurance points + Milestones

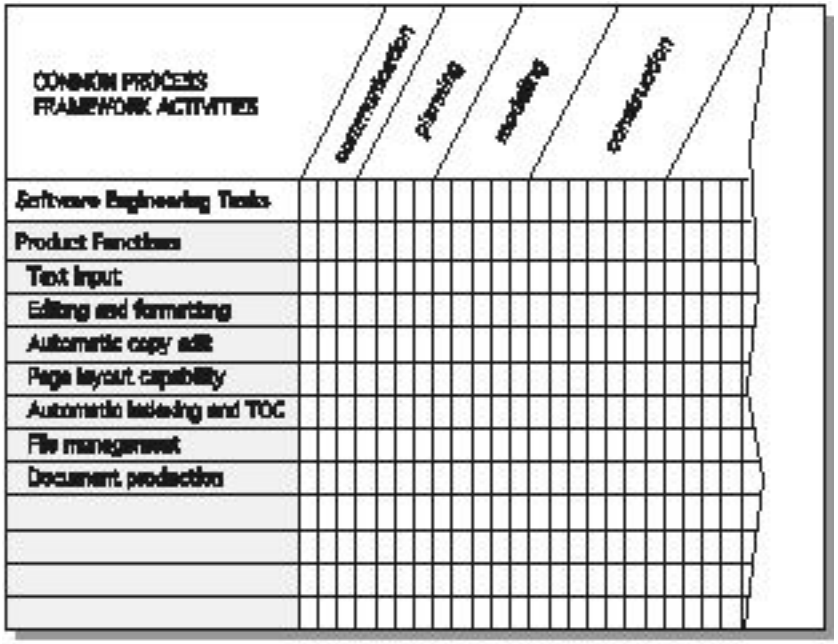


Figure 21.2 Melding the Problem and the Process

Common Sense Approach to Projects

Start on the right foot. This is accomplished by working hard (very hard) to understand the problem that is to be solved and then setting realistic objectives and expectations.

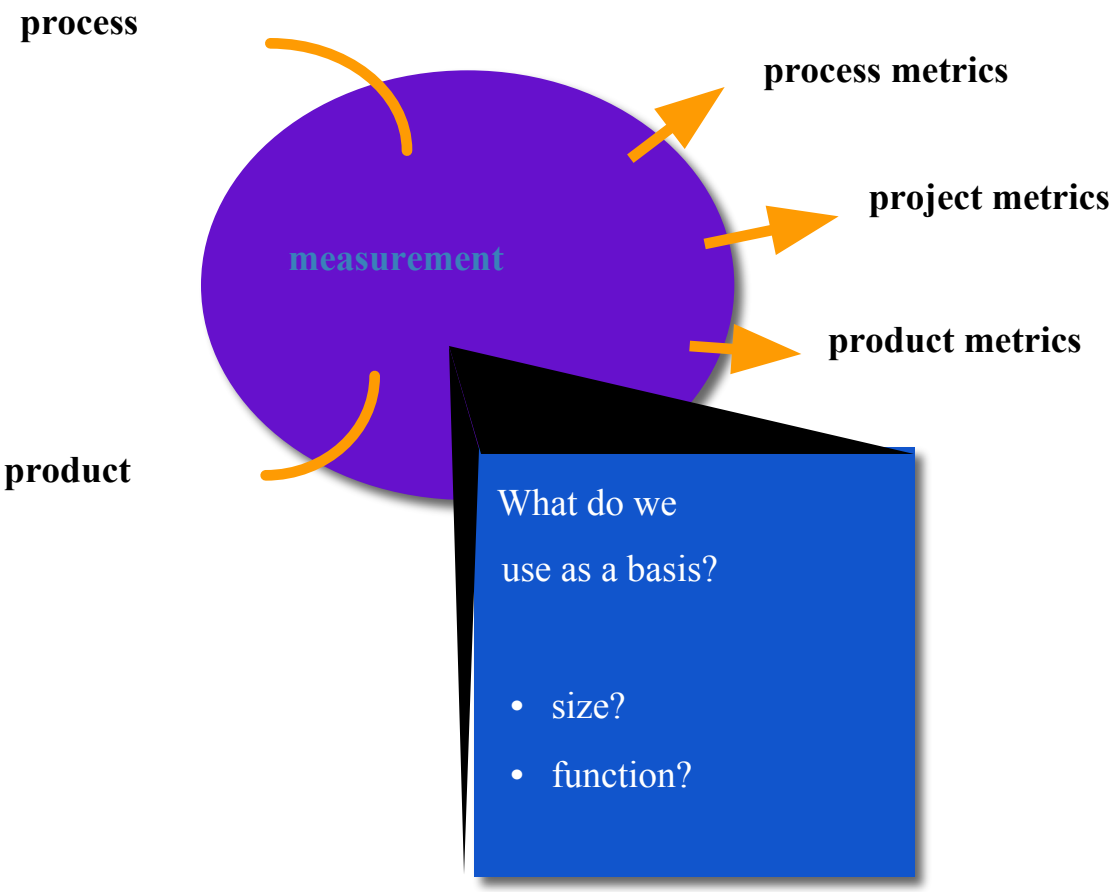
Maintain momentum. The project manager must provide incentives to keep turnover of personnel to an absolute minimum, the team should emphasize quality in every task it performs, and senior management should do everything possible to stay out of the team's way.

Track progress. For a software project, progress is tracked as work products (e.g., models, source code, sets of test cases) are produced and approved (using formal technical reviews) as part of a quality assurance activity.

Make smart decisions. In essence, the decisions of the project manager and the software team should be to "keep it simple."

Conduct a postmortem analysis. Establish a consistent mechanism for extracting lessons learned for each project.

Process and Project Metrics



Why do we measure?

- Assess the status of an ongoing project
- Track potential risks
- Uncover problem areas before they go “critical,”
- Adjust work flow or tasks,
- Evaluate the project team’s ability to control quality of software work products.

Process Measurement

We measure the efficacy of a software process indirectly be deriving set of metrics based on the outcomes that can be derived from the process.

Outcomes include
measures of errors uncovered before release of the software
defects delivered to and reported by end-users
work products delivered (productivity)
human effort expended
calendar time expended
schedule conformance
other measures.

Process Metric Guidelines

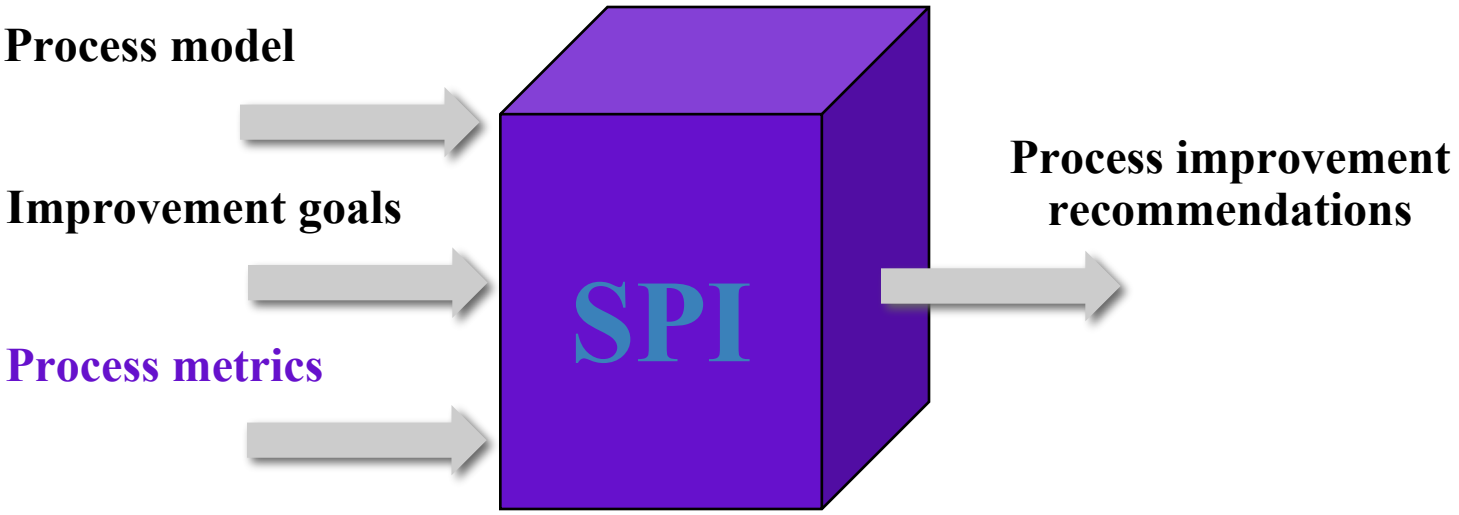
Use common sense and organizational sensitivity when interpreting metrics data.

Provide regular feedback to the individuals and teams who collect measures and metrics.

Don't use metrics to appraise individuals.
Work with practitioners and teams to set clear goals and metrics that will be used to achieve them.

Never use metrics to threaten individuals or teams.
Metrics data that indicate a problem area should not be considered "negative." These data are merely an indicator for process improvement.
Don't obsess on a single metric to the exclusion of other important metrics.

Software Process Improvement



Process Metrics

Quality-related

Focus on quality of work products and deliverables

Productivity-related

Production of work-products related to effort expended

Statistical SQA data

Error categorization & analysis

Defect removal efficiency

Propagation of errors from process activity to activity

Reuse data

The number of components produced and their degree of reusability

Project Metrics

Every project should measure:

Inputs—measures of the resources (e.g., people, tools) required to do the work.

Outputs—measures of the deliverables or work products created during the software engineering process.

Results—measures that indicate the effectiveness of the deliverables.

Typical Project Metrics

- Effort/time per software engineering task
- Errors uncovered per review hour
- Scheduled vs. actual milestone dates
- Changes (number) and their characteristics
- Distribution of effort on software engineering tasks

Size oriented metrics

1. Errors per KLOC (thousand lines of code)
2. Defects per KLOC
3. $ per LOC
4. Pages of documentation per KLOC
5. Errors per person-month
6. Errors per review hour
7. LOC per person-month
8. $ per page of documentation

Function Oriented Metrics

1. Errors per FP (thousand lines of code)
2. Defects per FP
3. $ per FP
4. Pages of documentation per FP
5. FP per person-month

Why opt for FP?

- Programming language independent
- Used readily countable characteristics that are determined early in the software process
- Does not “penalize” inventive (short) implementations that use fewer LOC that other more clumsy versions
- Makes it easier to measure the impact of reusable components

Object Oriented Metrics

- Number of scenario scripts (use-cases)
- Number of support classes (required to implement the system but are not immediately related to the problem domain)
- Average number of support classes per key class (analysis class)
- Number of subsystems (an aggregation of classes that support a function that is visible to the end-user of a system)

Establishing the Metric program

1. Identify your business goals.
2. Identify what you want to know or learn.
3. Identify your subgoals.
4. Identify the entities and attributes related to your subgoals.
5. Formalize your measurement goals.
6. Identify quantifiable questions and the related indicators that you will use to help you achieve your measurement goals.
7. Identify the data elements that you will collect to construct the indicators that help answer your questions.
8. Define the measures to be used, and make these definitions operational.
9. Identify the actions that you will take to implement the measures.
10. Prepare a plan for implementing the measures.

Example