RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL
New Scheme Based On AICTE Flexible Curricula
Computer Science and Engineering, VI-Semester
Open Elective - CS604 (B) Project Management

Topic Covered

Iterative process planning. Project organizations and responsibilities. Process automation. Project control and process instrumentation- core metrics, management indicators, life cycle expections, Process discriminants.

3.1. Iterative process planning

Iterative process planning is a procedure to make project. The whole project is not developed in only one increment the development of the project goes through the number of phases called increments. In iterative process the procedure is divided into the number of increments which is based on planning.

In each iteration plans are changed based on the feedback of outcome of the previous iteration given by the customers. It is a team based planning process the people who are doing the work must be involved in planning the project.

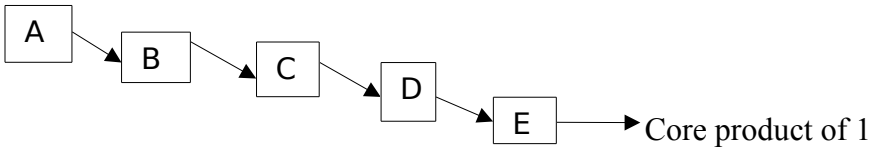
3.1.1 Working

- Take all the element of waterfall model and combine it then start working in iterative manner.
- Determine how many features can fit in each iteration.
- Make plan how to execute,
- Each iteration produces a core product and the core product consists of several numbers of features related to the project.
- The core product is deploying to the customers. Customers evaluate it and give feedback.
- Based on the feedback given by the customers second iteration is planned again a new core product is produced and so on.
- Repeat the process of iteration until the project was not developed.

3.1.2 Project functionality and feature

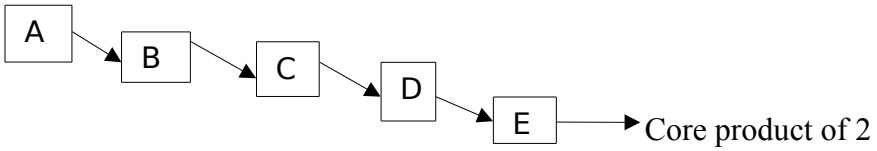
A. Planning
B. Design
C. Implementation
D. Testing
E. Deployment

Increment 1:



Delivery of increment 1-2

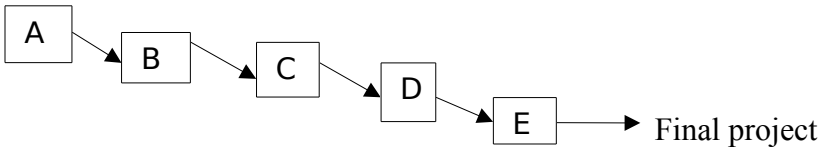
Increment 2:



Delivery of increment 2-----n

8

Increment n:



In each iteration the quality and the accuracy of the project are increased. Iterative process planning has linear and parallel work flow.

3.1.3 Advantages:

- Initial delivery cost is minimum
- Flexibility is high
- Changes done easily
- Risk analysis is better

3.1.4 Disadvantages:

- High total delivery cost.

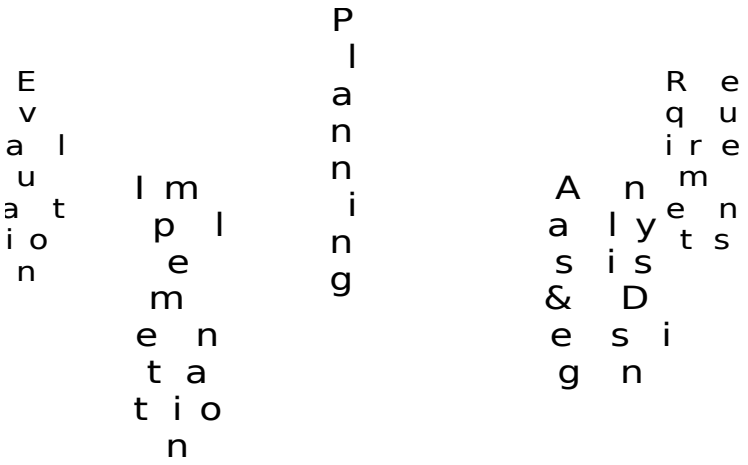


Figure: Iteration process

3. 2. Project organization and responsibilities

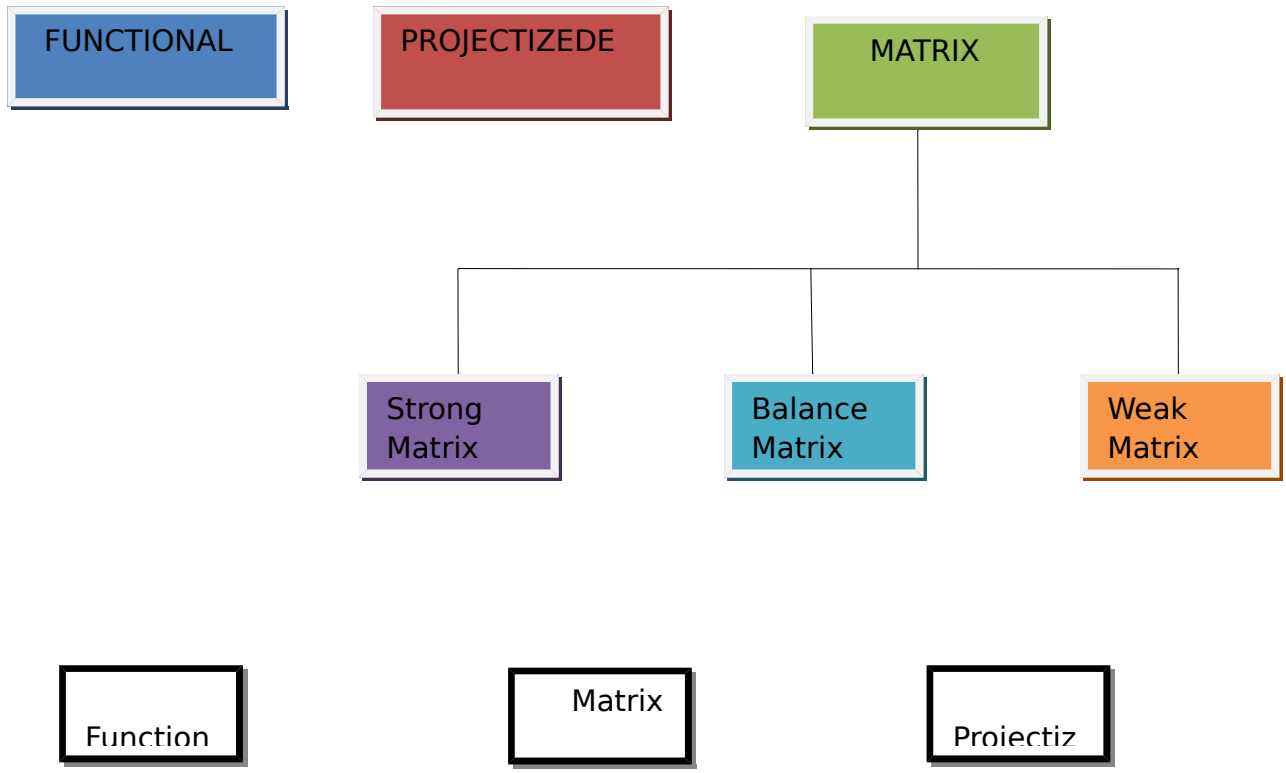
3.2.1 Project organization:

A project organization is a structure that simplifies the coordation and execution of projects activities. The main reason behind this is to create a surrounding that provide interaction among the team and minimize interruption, difficulties and overlaps while doing project. Structure of any project is created under project organisation.

3.2.1.1 Organization of any project is influenced by two ways

1. Authority of project manager.
2. The way in which projects are conducted.

3.2.1.2 Organisation is structured in one of the three ways:



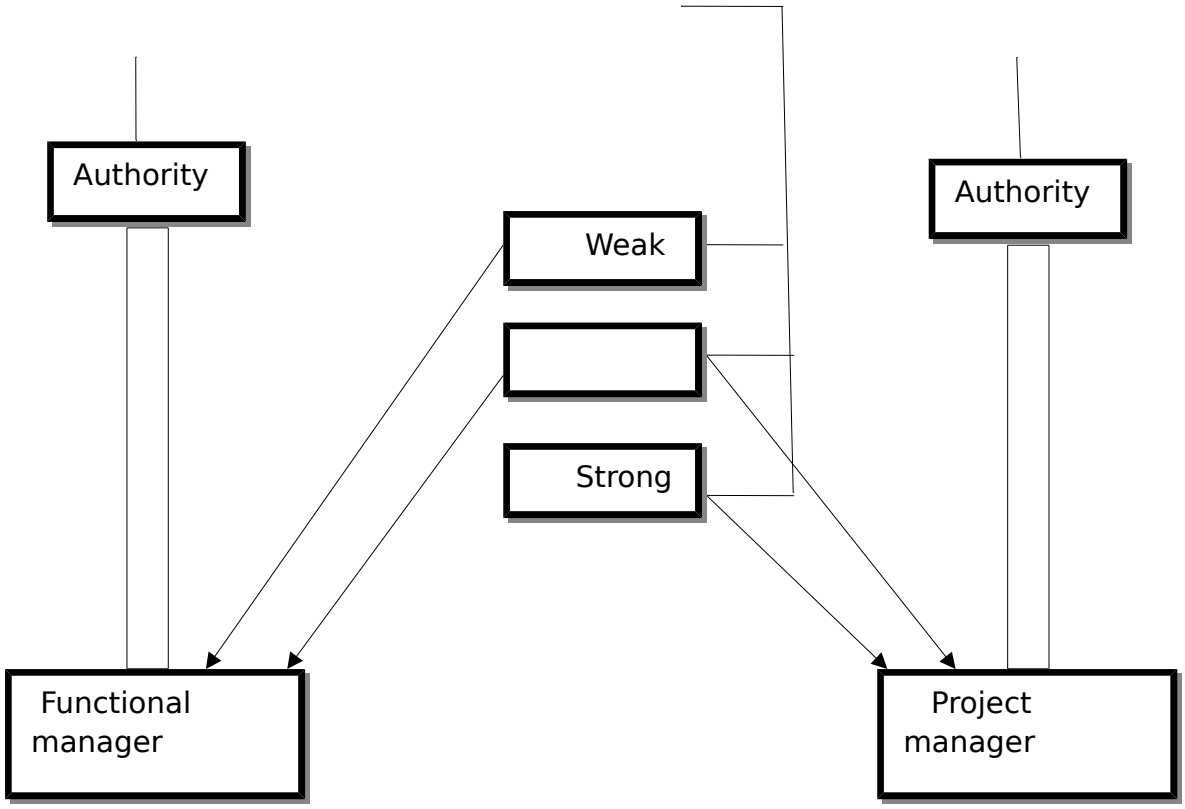


Figure 3: Project organization structure

3.2.2. FUNCTIONAL ORGANIZATION:

In functional organization people are grouped into their area of specialization as supervised in functional manager where their expertise in required field. Functional organisation is based on team work like marketing, finance, operation etc. Each team has a team leader, who supervised the team.

3.2.3. PROJECTIZED ORGANISATION:

Projectized organisation is opposite of functional organisation. In projectized organisation project is organized by itself. Main idea is to develop loyalty to the project. Project manager has all the authority and power to deal with the project. Communication between the project manager and team member is strong with single reporting system.

3.2.4 MATRIX:

Matrix organization is used to correlate the functional manager and the project manager. It is used to minimize the differences between functional and projectized organization

Matrix organization can be classified into three types. They are:

1. Weak matrix

2. Balanced matrix

3. Strong matrix

WEAK MATRIX:

In weak matrix project manager has limited number of authority and power. It belongs to the class of functional organization. The role of project manager is part time.

BALANCED MATRIX:

In balanced matrix power and authority is equally shared between functional manager and project manager. 60% of project comes under this matrix. The role of project manager and functional manager is equal.

STRONG MATRIX:

Project manager have all the authority to make decision of concerned project. It is a class of projectized organization. The role of project manager is full time.

Merits:

- Focus on project by project manager.
- Management of time is excellent.

Demerits:

- Communication gap between team members.
- Low coordination.
- Insecurity.

Responsibilities:

- Responsibility of project manager is to lead the project.
- Achieving the goal of the project.
- Manage process of whole project.
- Provide tools and technique that are required in the project.
- Planning and organization of work.
- Manage day to day activities and check the progress of the project time to time.
- Delivered the project to the customers on the given time period.

References:

- http://webcache.googleusercontent.com/search?q=cache:0rwC50tV8CYJ:https://wenku.baidu.com/view/25999e1af18583d0496459bc.html?re%3Dview&hl=en&gl=in&strip=1&vwsrc=0
- https://en.wikipedia.org/wiki/Project_management
- https://www.pm4dev.com/pm4dev-blog/entry/what-is-iterative-planning.html
- https://www.gristprojectmanagement.us/software-3/iterative-process-planning.html

3.3 Project Control & Instrumentation

INTERODUCTION: Software metrics are used to implement the activities and products of the software development process. Hence, the quality of the software products and the achievements in the development process can be determined using the software metrics.

Need for Software Metrics:

- Software metrics are needed for calculating the cost and schedule of a software product with great accuracy.
- Software metrics are required for making an accurate estimation of the progress.
- The metrics are also required for understanding the quality of the software product.

3.3.1 INDICATORS:

An indicator is a metric or a group of metrics that provides an understanding of the software process or software product or a software project. A software engineer assembles measures and produce metrics from which the indicators can be derived. Two types of indicators are:

(i) Management indicators.

(ii) Quality indicators.

3.3.1.1 Management Indicators

The management indicators i.e., technical progress, financial status and staffing progress are used to determine whether a project is on budget and on schedule. The management indicators that indicate financial status are based on earned value system.

3.3.1.2 Quality Indicators

The quality indicators are based on the measurement of the changes occurred in software.

3.3.2 SEVEN CORE METRICS OF SOFTWARE PROJECT

Software metrics instrument the activities and products of the software development/integration process. Metrics values provide an important perspective for

managing the process. The most useful metrics are extracted directly from the evolving artefacts.

There are seven core metrics that are used in managing a modern process.

Seven core metrics related to project control:

Management Indicators	Quality Indicators
Work and Progress	Change traffic and stability
Budgeted cost and expenditures	Breakage and modularity
Staffing and team dynamics	Rework and adaptability
	Mean time between failures (MTBF) and maturity

3.3.2.1 MANAGEMENT INDICATORS:

1. Work and progress

This metric measure the work performed over time. Work is the effort to be accomplished to complete a certain set of tasks. The various activities of an iterative development project can be measured by defining a planned estimate of the work in an objective measure, then tracking progress (work completed overtime) against that plan.

The default perspectives of this metric are:

Software architecture team: - Use cases demonstrated.

Software development team: - SLOC under baseline change management, SCOs closed

Software assessment team: - SCOs opened, test hours executed and evaluation criteria meet.

Software management team: - milestones completed.

The below figure shows expected progress for a typical project with three major releases

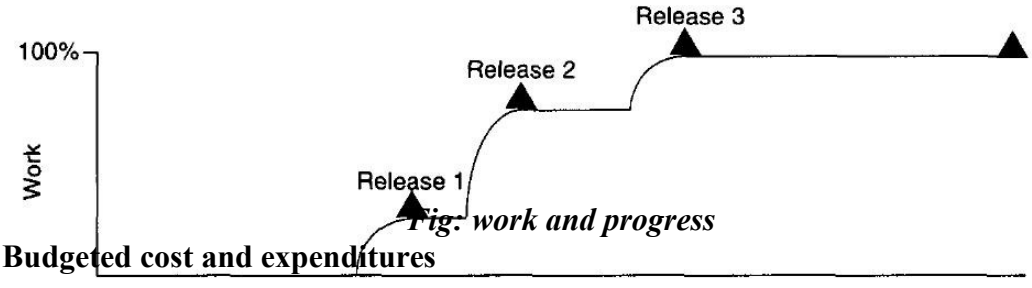


Fig: work and progress

2. Budgeted cost and expenditures

This metric measures cost incurred over time. Budgeted cost is the planned expenditure profile over the life cycle of the project. To maintain management control, measuring cost expenditures over the project life cycle is always necessary. Tracking financial progress takes on an organization - specific format. Financial performance can be measured by the use of an earned value system, which provides highly detailed cost and

schedule insight. The basic parameters of an earned value system, expressed in units of dollars, are as follows:

Expenditure Plan - It is the planned spending profile for a project over its planned schedule.

Actual progress - It is the technical accomplishment relative to the planned progress underlying the spending profile.

Actual cost: It is the actual spending profile for a project over its actual schedule.

Earned value: It is the value that represents the planned cost of the actual progress.

Cost variance: It is the difference between the actual cost and the earned value.

Schedule variance: It is the difference between the planned cost and the earned value.

Of all parameters in an earned value system, actual progress is the most subjective

Assessment: Because most managers know exactly how much cost they have incurred and how much schedule they have used, the variability in making accurate assessments is centered in the actual progress assessment. The default perspectives of this metric are cost per month, full-time staff per month and percentage of budget expended.

3. Staffing and team dynamics

This metric measure the personnel changes over time, which involves staffing additions and reductions over time. An iterative development should start with a small team until the risks in the requirements and architecture have been suitably resolved. Depending on the overlap of iterations and other project specific circumstances, staffing can vary. Increase in staff can slow overall project progress as new people consume the productive team of existing people in coming up to speed. Low attrition of good people is a sign of success. The default perspectives of this metric are people per month added and people per month leaving.

These three management indicators are responsible for technical progress, financial status and staffing progress.

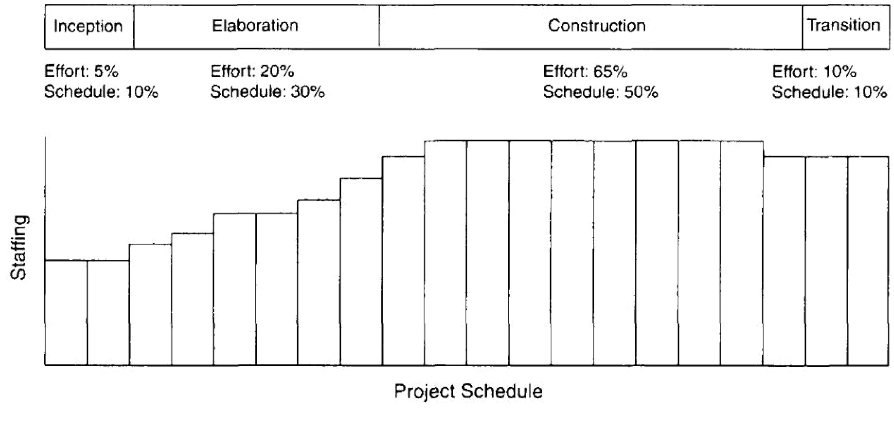


Fig: staffing and Team dynamics

3.3.2.2 QUALITY INDICATORS:

1. **Change traffic and stability:**

This metric measures the change traffic over time. The number of software change orders opened and closed over the life cycle is called change traffic. Stability specifies the relationship between opened versus closed software change orders. This metric can be collected by change type, by release, across all releases, by term, by components, by subsystems, etc.

The below figure shows stability expectation over a healthy project's life cycle

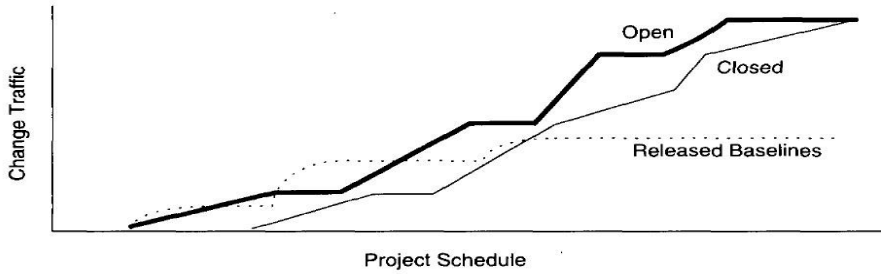


Fig: Change traffic and stability

2. **Breakage and modularity**

This metric measures the average breakage per change over time. Breakage is defined as the average extent of change, which is the amount of software baseline that needs rework and measured in source lines of code, function points, components, subsystems, files or other units.

Modularity is the average breakage trend over time. This metric can be collected by revoke SLOC per change, by change type, by release, by components and by subsystems.

3. **Rework and adaptability:**

This metric measures the average rework per change over time. Rework is defined as the average cost of change which is the effort to analyze, resolve and retest all changes to software baselines. Adaptability is defined as the rework trend over time. This metric provides insight into rework measurement. All changes are not created equal. Some changes

can be made in a staff- hour, while others take staff-weeks. This metric can be collected by average hours per change, by change type, by release, by components and by subsystems.

4. ***MTBF* and Maturity:**

This metric measures defect rater over time. MTBF (Mean Time Between Failures) is the average usage time between software faults. It is computed by dividing the test hours by the number of type 0 and type 1 SCOs. Maturity is defined as the MTBF trend over time.

Software errors can be categorized into two types deterministic and nondeterministic. Deterministic errors are also known as Bohr-bugs and nondeterministic errors are also called as Heisen-bugs. Bohr-bugs are a class of errors caused when the software is stimulated in a certain way such as coding errors. Heisen-bugs are software faults that are coincidental with a certain probabilistic occurrence of a given situation, such as design errors. This metric can be collected by failure counts, test hours until failure, by release, by components and by subsystems. These four quality indicators are based primarily on the measurement of software change across evolving baselines of engineering data.

3.4 METRICS AUTOMATION:

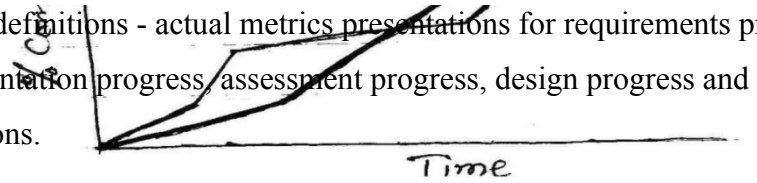
Many opportunities are available to automate the project control activities of a software project. A Software Project Control Panel (SPCP) is essential for managing against a plan. This panel integrates data from multiple sources to show the current status of some aspect of the project. The panel can support standard features and provide extensive capability for detailed situation analysis. SPCP is one example of metrics automation approach that collects, organizes and reports values and trends extracted directly from the evolving engineering artifacts.

SPCP:

To implement a complete SPCP, the following are necessary.

- Metrics primitives - trends, comparisons and progressions
- A graphical user interface.
- Metrics collection agents
- Metrics data management server

- Metrics definitions - actual metrics presentations for requirements progress, implementation progress, assessment progress, design progress and other progress dimensions.



- Actors - monitor and administrator.

Monitor defines panel layouts, graphical objects and linkages to project data. Specific monitors called roles include software project managers, software development team leads, software architects and customers. Administrator installs the system, defines new mechanisms, graphical objects and linkages. The whole display is called a panel. Within a panel are graphical objects, which are types of layouts such as dials and bar charts for information. Each graphical object displays a metric. Apanel contains a number of graphical objects positioned in a particular geometric layout. A metric shown in a graphical object is labeled with the metric type, summary level and insurance name (line of code, subsystem, server1). Metrics can be displayed in two modes – value, referring to a given point in time and graph referring to multiple and consecutive points in time. Metrics can be displayed with or without control values. A control value is an existing expectation either absolute or relative that is used for comparison with a dynamically changing metric. Thresholds are examples of control values.

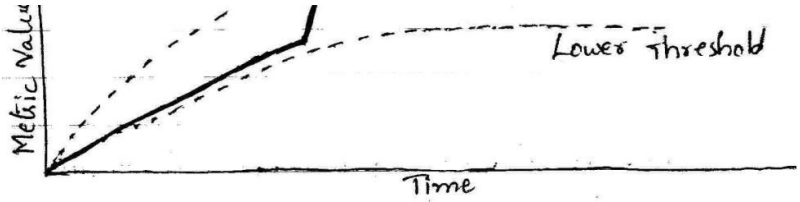


Figure*:-*The basic fundamental metrics classes are trend, comparison and progress.

The format and content of any project panel are configurable to the software project manager's preference for tracking metrics of top-level interest. The basic operation of an SPCP can be described by the following top -level use case.

i. Start the SPCP

ii. Select a panel preference
iii. Select a value or graph metric
iv. Select to superimpose controls

v. Drill down to trend
vi. Drill down to point in time.
vii. Drill down to lower levels of information
viii. Drill down to lower level of indicators.

Reference:

- Rajeev Mall, Book Software Engineering
- Roger Presman Book.
- http://www.pvpsiddhartha.ac.in/dep_it/lecture%20notes/SPM/unit5.pdf

3.5 Project Life Cycle Expectation

There is no mathematical or formal derivation for using the seven core metrics. However, there were specific reasons for selecting them:

• The quality indicators are derived from the evolving product rather than from the artifacts.

• They provide insight into the waste generated by the process. Scrap and rework metrics are a standard measurement perspective of most manufacturing processes.

• They recognize the inherently dynamic nature of an iterative development process. Rather than focus on the value, they explicitly concentrate on the trends or changes with respect to time.

• The combination of insight from the current value and the current trend provides tangible indicators for management action.

The actual values of these metrics can vary widely across projects, organizations, and domains. The relative trends across the project phases, however, should follow the general pattern shown in Table. A mature development organization should be able to describe metrics targets that are much more definitive and precise for its line of business and specific processes.

Table. The default pattern of life-cycle metrics evolution

Metric	inception	elaboration	construction	transition

Progress	5%	25%	90%	100%
Architecture	30%	90%	100%	100%
Applications	<5%	20%	85%	100%
Expenditures	Low	Moderate	High	High
Effort	5%	25%	90%	100%
Schedule	10%	40%	90%	100%
Staffing	Small team	Ramp up	Steady	Varying
Stability	Volatile	Moderate	Moderate	Stable
Architecture	Volatile	Moderate	Stable	Stable
Applications	Volatile	Volatile	Moderate	Stable
Modularity	50%-100%	25%-50%	<25%	5%-10%
Architecture	>50%	>50%	<15%	<5%
Applications	>80%	>80%	<25%	<10%
Adaptability	Varying	Varying	Benign.	Benign
Architecture	Varying	Moderate	Benign	Benign
Applications	Varying	Varying	Moderate	Benign
Maturity	Prototype	Fragile	Usable	Robust
Architecture	Prototype	Usable	Robust	Robust
Applications	Prototype	Fragile	Usable	Robust

3.6 Process Discriminants

Introduction:

In tailoring the management process to a specific domain or project, there are two dimensions of discriminating factors: technical complexity and management complexity. A process

framework is not a project-specific process implementation with a well-defined recipe for success. The process framework must be configured to the specific characteristics of the project. The process discriminants are organized around six process parameters - scale, stakeholder cohesion, process flexibility, process maturity, architectural risk and domain experience.

1. **Scale**: the scale of the project is the team size, which drives the process configuration more than any other factor. There are many ways to measure scale, including number of sources lines of code, number of function points, number of use cases and number of dollars. The primary measure of scale is the size of the team. Five people are an optimal size for an engineering team. A team consisting of 1 member is said to be trivial, a team of 5 is said to be small, a team of 25 is said to be moderate, a team of 125 is said to be large, a team of 625 is said to be huge and so on. As team size grows, a new level of personnel management is introduced at each factor of 5.

Trivial - sized projects require almost no management overhead. Only little documentation is required. Workflow is single-threaded. Performance is dependent on personnel skills.Small projects consisting of 5 people require very little management overhead. Project milestones are easily planned, informally conducted and changed. There are a small number of individual workflows. Performance depends primarily on personnel skills. Process maturity is unimportant.

Moderate-sized projects consisting of 25 people require very moderate management overhead. Project milestones are formally planned and conducted. There are a small number of concurrent team workflows, each team consisting of multiple individual workflows. Performance is highly dependent on the skills of key personnel. Process maturity is valuable.

Large projects consisting of 125 people require substantial management overhead. Project milestones are formally planned and conducted. A large number of concurrent team workflows are necessary, each with multiple individual workflows. Performance is highly dependent on the skills of the key personnel. Process maturity is necessary.

Huge projects consisting of 625 people require substantial management overhead. Project milestones are very formally planned and conducted. There are a very large number of concurrent team workflows, each with multiple individual workflows. Performance is highly

dependent on the skills of the key personnel. Project performance is still dependent on average people.

2. **Stakeholder Cohesion or Contention:** The degree of cooperation and coordination among stakeholders (buyers, developers, users, subcontractors and maintainers) significantly drives the specifics of how a process is defined. This process parameter ranges from cohesive to adversarial. Cohesive teams have common goals, complementary skills and close communications. Adversarial teams have conflicting goals, competing and incomplete skills, and less-than-open communication.

3. **Process Flexibility or Rigor**: The implementation of the project's process depends on the degree of rigor, formality and change freedom evolved from projects contract (vision document, business case and development plan). For very loose contracts such as building a commercial product within a business unit of a software company, management complexity is minimal. For a very rigorous contract, it could take many months to authorize a change in a release schedule.

4. **Process Maturity:** The process maturity level of the development organization is the key driver of management complexity. Managing a mature process is very simpler than managing an immature process. Organization with a mature process have a high level of precedent experience in developing software and a high level of existing process collateral that enables predictable planning and execution of the process. This sort of collateral includes well-defined methods, process automation tools, and trained personnel, planning metrics, artifact templates and workflow templates.

5. **Architectural Risk:** The degree of technical feasibility is an important dimension of defining a specific projects process. There are many sources of architecture risk. They are (1) system performance which includes resource utilization, response time, throughout and accuracy, (2) robustness to change which includes addition of new features & incorporation of new technology and (3) system reliability which includes predictable behaviour and fault tolerance.

6**. Domain Experience:** The development organization's domain experience governs its ability to converge on an acceptable architecture in a minimum no of iterations.

Reference:

- **Error! Bookmark not defined.**

http://www.crectirupati.com/sites/default/files/lecture_notes/spm%20notes.pdf