UNIT - IV

Project Organizations and Responsibilities: Line-of-Business Organizations, Project Organizations, evolutionof Organizations.

Process Automation: Automation Building blocks, The Project Environment.

Project Organizations and Responsibilities:

- **Organizations** engaged in software Line-of-Business need to support projects with the infrastructurenecessary to use a common process.
- **Project** organizations need to allocate artifacts & responsibilities across project team to ensure a balanceof global (architecture) & local (component) concerns.
- **The organization** must evolve with the WBS & Life cycle concerns.
- **Software lines of business & product teams have different motivation.**
- **Software lines of business** are motivated by return of investment (ROI), new business discriminators, market diversification & profitability.
- **Project teams** are motivated by the cost, Schedule & quality of specific deliverables

1) Line-Of-Business Organizations:

The main features of default organization are as follows:

- Responsibility for process definition & maintenance is specific to a cohesive line of business.
- Responsibility for process automation is an organizational role & is equal in importance to theprocess definition role.
- Organizational role may be fulfilled by a single individual or several different teams.

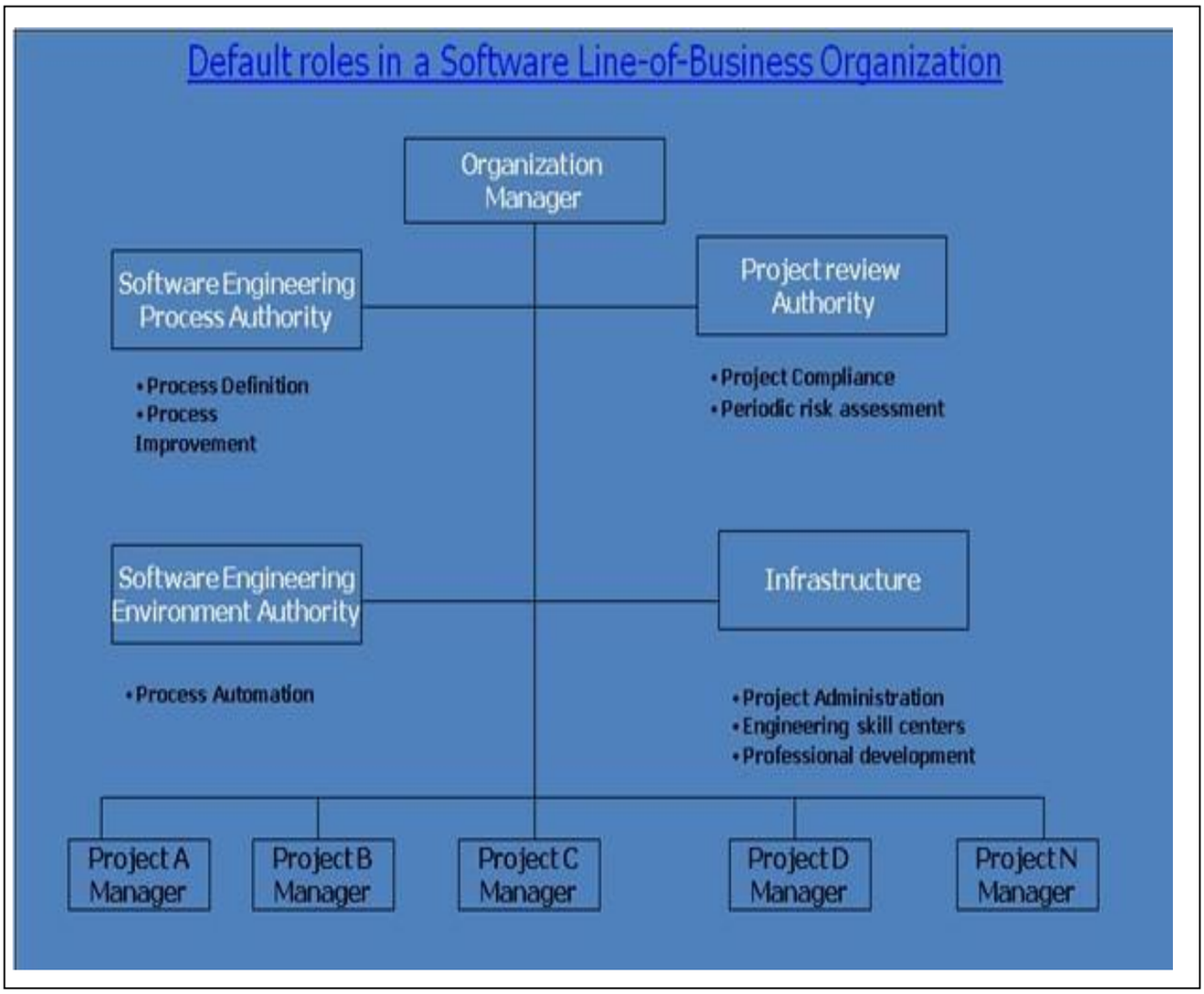


Fig: Default roles in a software Line-of-Business Organization.

Software Engineering Process Authority (SEPA)

The SEPA facilities the exchange of information & process guidance both to & from projectpractitioners

This role is accountable to General Manager for maintaining a currentassessment of theorganization's process maturity & its plan for future improvement

Project Review Authority (PRA)

The PRA is the single individual responsible for ensuring that a software project complies withall organizational & business unit software policies, practices & standards

A software Project Manager is responsible for meeting the requirements of a contract or some otherproject compliance standard

Software Engineering Environment Authority(SEEA)

The SEEA is responsible for automating the organization's process, maintaining the organization'sstandard environment, Training projects to use the environment & maintaining organization-wide reusable assets

The SEEA role is necessary to achieve a significant ROI for common process.

Infrastructure

An organization's infrastructure provides human resources support, project-independentresearch & development, & other capital software engineering assets.

2) Project organizations:

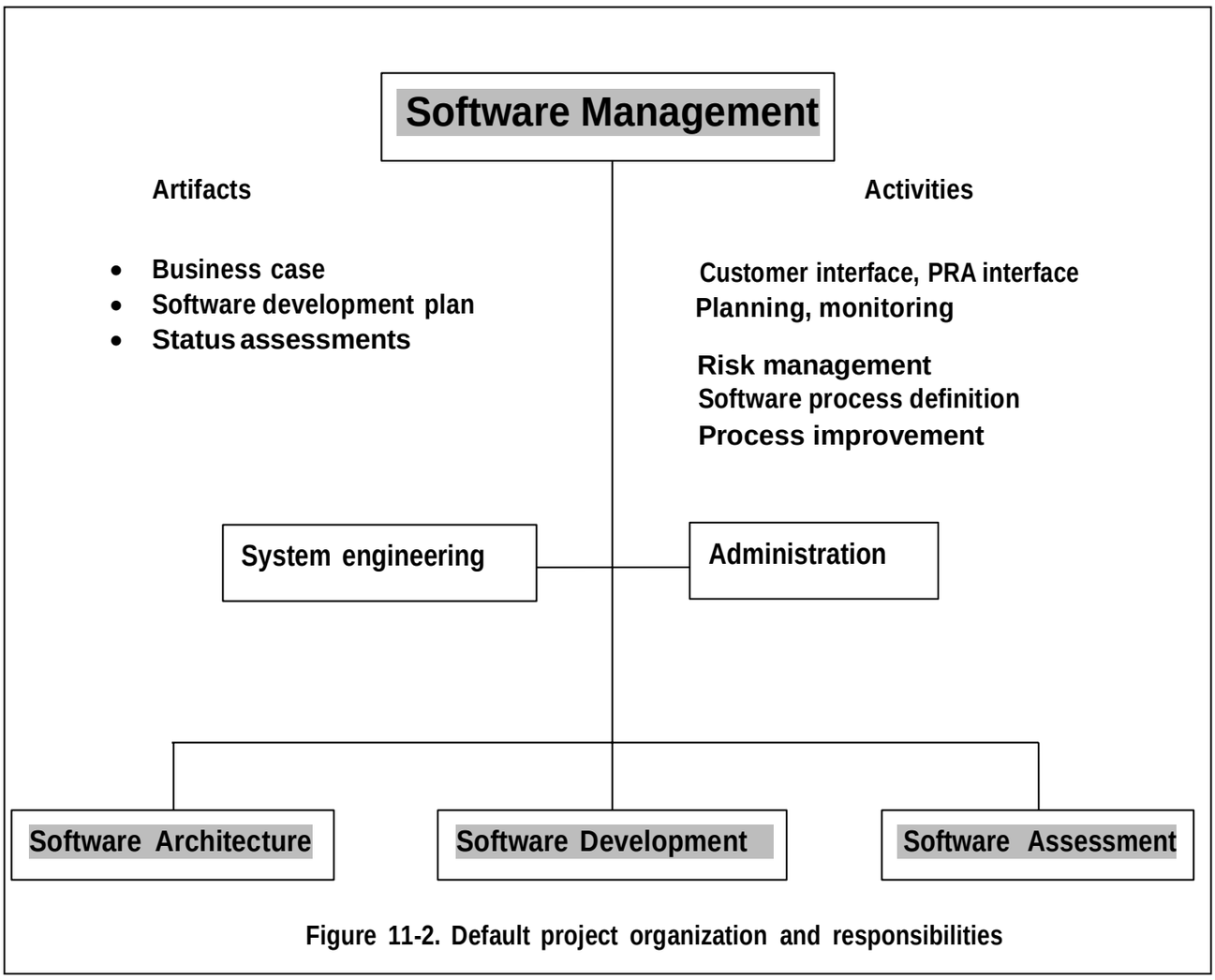
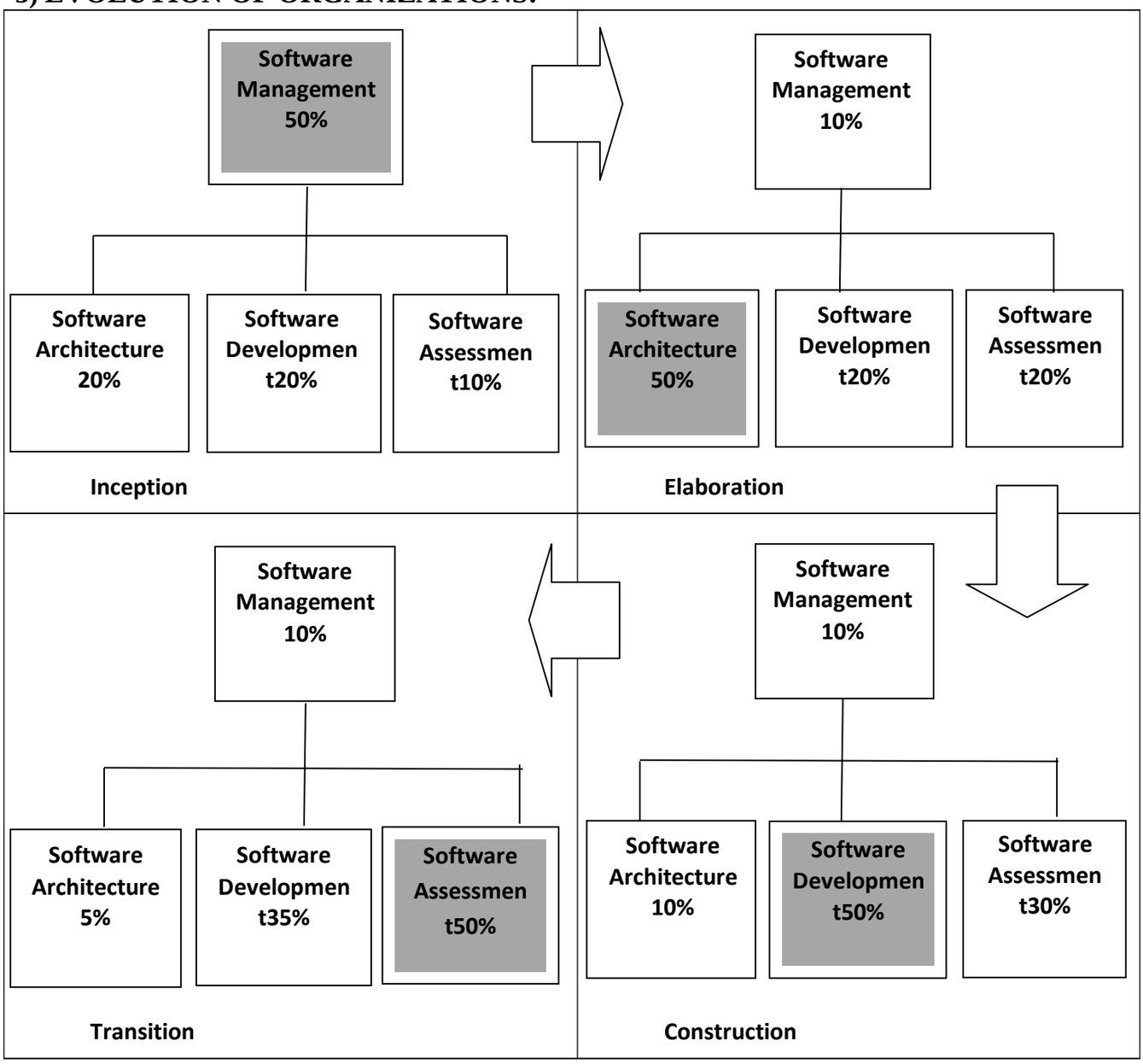


Figure 11-2. Default project organization and responsibilities

- The above figure shows a default project organization and maps project-level roles andresponsibilities.
- The main features of the default organization are as follows:
- **The project management team** is an active participant, responsible for producing as well asmanaging.

- **The architecture team** is responsible for real artifacts and for the integration of components,not just for staff functions.
- **The development team** owns the component construction and maintenance activities.
- The assessment team is separate from development.
- **Quality** is everyone's into all activities and checkpoints.
- Each team takes responsibility for a different quality perspective.

3) EVOLUTION OF ORGANIZATIONS:



Inception: Software management: **50%** Software Architecture:20% Software development: 20% Software Assessment (measurement/evaluation):10%	**Elaboration:** Software management: 10% Software Architecture:**50%** Software development: 20% Software Assessment (measurement/evaluation):20%
Construction: Software management: 10% Software Architecture:10% Software development: **50%** Software Assessment (measurement/evaluation):30%	**Transition:** Software management: 10% Software Architecture: 5% Software development: 35% Software Assessment (measurement/evaluation):**50%**

The Process Automation:

Introductory

Remarks:

The environment must be the first-class artifact of the process.

Process automation & change management is critical to an iterative process. If the change is expensive thenthe development organization will resist it.

Round-trip engineering & integrated environments promote change freedom & effective evolution oftechnical artifacts.

Metric automation is crucial to effective project control.

External stakeholders need access to environment resources to improve interaction with the development team& add value to the process.

The three levels of process which requires a certain degree of process automation for the corresponding processto be carried out efficiently.

Metaprocess (Line of business): The automation support for this level is called an infrastructure. **Macroproces (project):** The automation support for a project's process is called an environment. **Microprocess (iteration):** The automation support for generating artifacts is generally called a tool.

Tools: Automation Building blocks:

Many tools are available to automate the software development process. Most of the core softwaredevelopment tools map closely to one of the process workflows

Workflows	Environment Tools & process Automation
Management automation	Workflow automation, Metrics
Environment	Change Management, Document
Automation	
Requirements	Requirement Management
Design	Visual Modeling
Implementation	-Editors, Compilers, Debugger, Linker,
RuntimeAssessment	-Test automation, defect
Tracking	
Deployment	defect Tracking

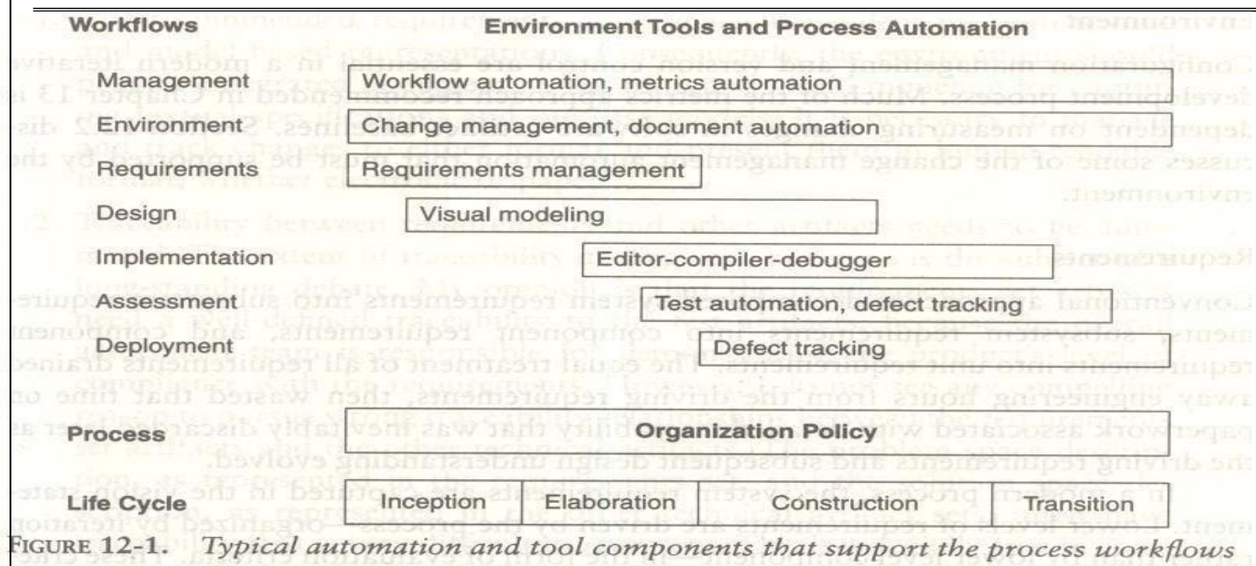


FIGURE 12-1. *Typical automation and tool components that support the process workflows*

The Project Environment:

The project environment artifacts evolve through three discrete states.
(1) Prototyping Environment. (2) Development Environment. (3) Maintenance Environment.
The **Prototype Environment** includes an architecture test bed for prototyping project architecture to evaluatetrade-offs during inception & elaboration phase of the life cycle.
The **Development environment** should include a full suite of development tools needed to support variousProcess workflows & round-trip engineering to the maximum extent possible.
The Maintenance Environment should typically coincide with the mature version of the development. There are four important environment disciplines that are critical to management context & the success of amodern iterative development process.

Round-Trip engineering

Change Management

Software Change Orders

(SCO)

Configuration baseline Configuration Control Board

Infrastructure

Organization Policy

Organization

Environment

Stakeholder

Environment.

Round Trip Environment

Tools must be integrated to maintain consistency & traceability.
Round-Trip engineering is the term used to describe this key requirement for environment that support iterativedevelopment.
As the software industry moves into maintaining different information sets for the engineering artifacts, moreautomation support is needed to ensure efficient & error free transition of data from one artifacts to another. Round-trip engineering is the environment support necessary to maintain Consistency among the engineeringartifacts.

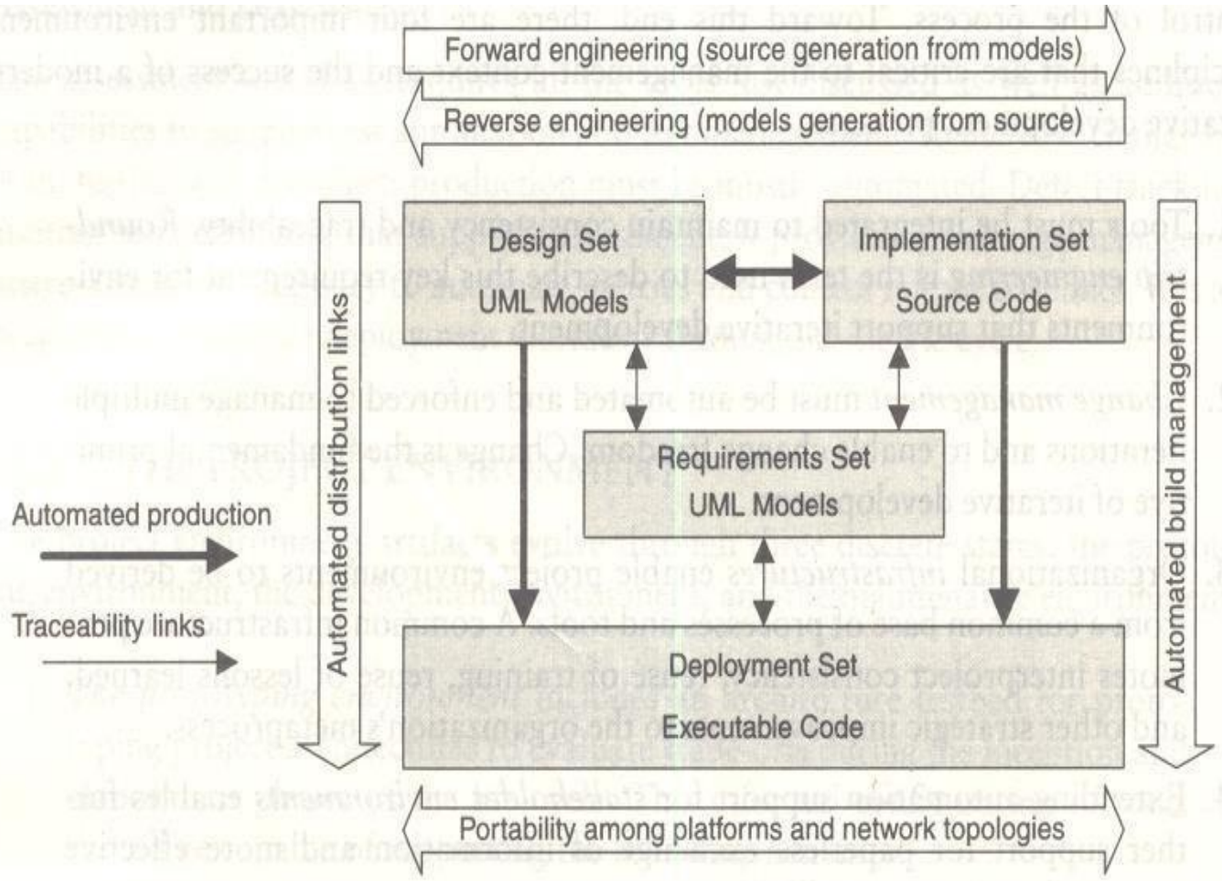


FIGURE 12-2. *Round-trip engineering*

Change Management

Change management must be automated & enforced to manage multiple iterations & to enable change freedom.Change is the fundamental primitive of iterative Development.

I. Software Change Orders

The atomic unit of software work that is authorized to create, modify or obsolesce components within aconfiguration baseline is called a software change orders (SCO)

The basic fields of the SCO are Title, description, metrics, resolution, assessment & disposition

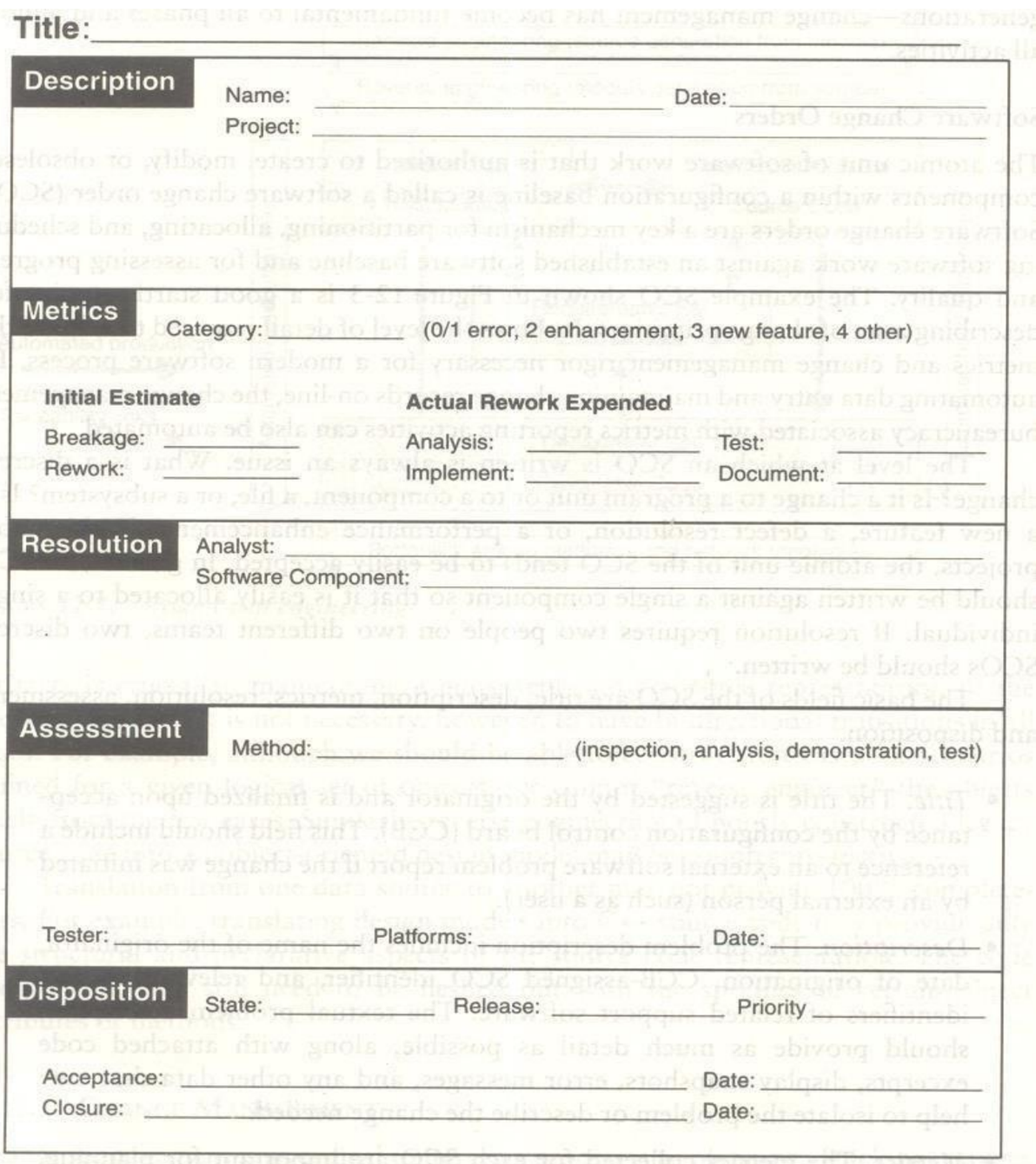


FIGURE 12-3. *The primitive components of a software change order*

Change management

II. Configuration Baseline

A configuration baseline is a named collection of software components &Supporting documentation that is subjected to change management & is upgraded, maintained, tested, statuses & obsolesced a unitThere are generally two classes of baselines

External Product
ReleaseInternal testing
Release

Three levels of baseline releases are required for most Systems

1. Major release (N)
2. Minor Release (M)
3. Interim (temporary) Release (X)

Major release represents a new generation of the product or project

A minor release represents the same basic product but with enhanced features, performance or quality.**Major & Minor** releases are intended to be external product releases that are persistent & supported for a period of time.

An interim release corresponds to a developmental configuration that is intended to be transient. Once software is placed in a controlled baseline all changes are tracked such that a distinction must bemade for the cause of the change. Change categories are

Type 0: Critical Failures (must be fixed before release)

Type 1: A bug or defect either does not impair (Harm) the usefulness of the system or can be workedaround

Type 2: A change that is an enhancement rather than a response to a defect**Type** 3: A change that is necessitated by the update to the environment **Type** 4: Changes that are not accommodated by the other categories.

Change Management

III Configuration Control Board (CCB)

A CCB is a team of people that functions as the decisionAuthority on the content of configuration baselines

A CCB includes:

1. Software managers
2. Software Architecture managers
3. Software Development managers
4. Software Assessment managers
5. Other Stakeholders who are integral to the maintenance of the controlled software deliverysystem?

Infrastructure

The organization infrastructure provides the organization's capital assets including two keyartifacts - Policy & Environment

I Organization Policy:

A Policy captures the standards for project software development processes

The organization policy is usually packaged as a handbook that defines the life cycles & the process primitives such as

- Major milestones
- Intermediate Artifacts
- Engineering repositories
- Metrics
- Roles & Responsibilities

I. **Process-primitive definitions**
   A. Life-cycle phases (inception, elaboration, construction, transition)
   B. Checkpoints (major milestones, minor milestones, status assessments)
   C. Artifacts (requirements, design, implementation, deployment, management sets)
   D. Roles and responsibilities (PRA, SEPA, SEEA, project teams)

II. **Organizational software policies**
   A. Work breakdown structure
   B. Software development plan
   C. Baseline change management
   D. Software metrics
   E. Development environment
   F. Evaluation criteria and acceptance criteria
   G. Risk management
   H. Testing and assessment

III. **Waiver policy**

IV. **Appendixes**
   A. Current process assessment
   B. Software process improvement plan

FIGURE 12-5. *Organization policy outline*

Infrastructure

II Organization Environment

The Environment that captures an inventory of tools which are building blocks from which project environments can be configured efficiently & economically

Stakeholder Environment

Many large scale projects include people in external organizations that represent other stakeholdersparticipating in the development process they might include

- Procurement agency contract monitors
- End-user engineering support personnel
- Third party maintenance contractors
- Independent verification & validation contractors
- Representatives of regulatory agencies & others.

These stakeholder representatives also need to access to development resources so that they cancontribute value to overall effort. These stakeholders will be access through on-line

An on-line environment accessible by the external stakeholders allow them to participate in the processa follows

Accept & use executable increments for the hands-on evaluation.

Use the same on-line tools, data & reports that the development organization uses to manage &monitor the project

Avoid excessive travel, paper interchange delays, format translations, paper * shipping costs & otheroverhead cost

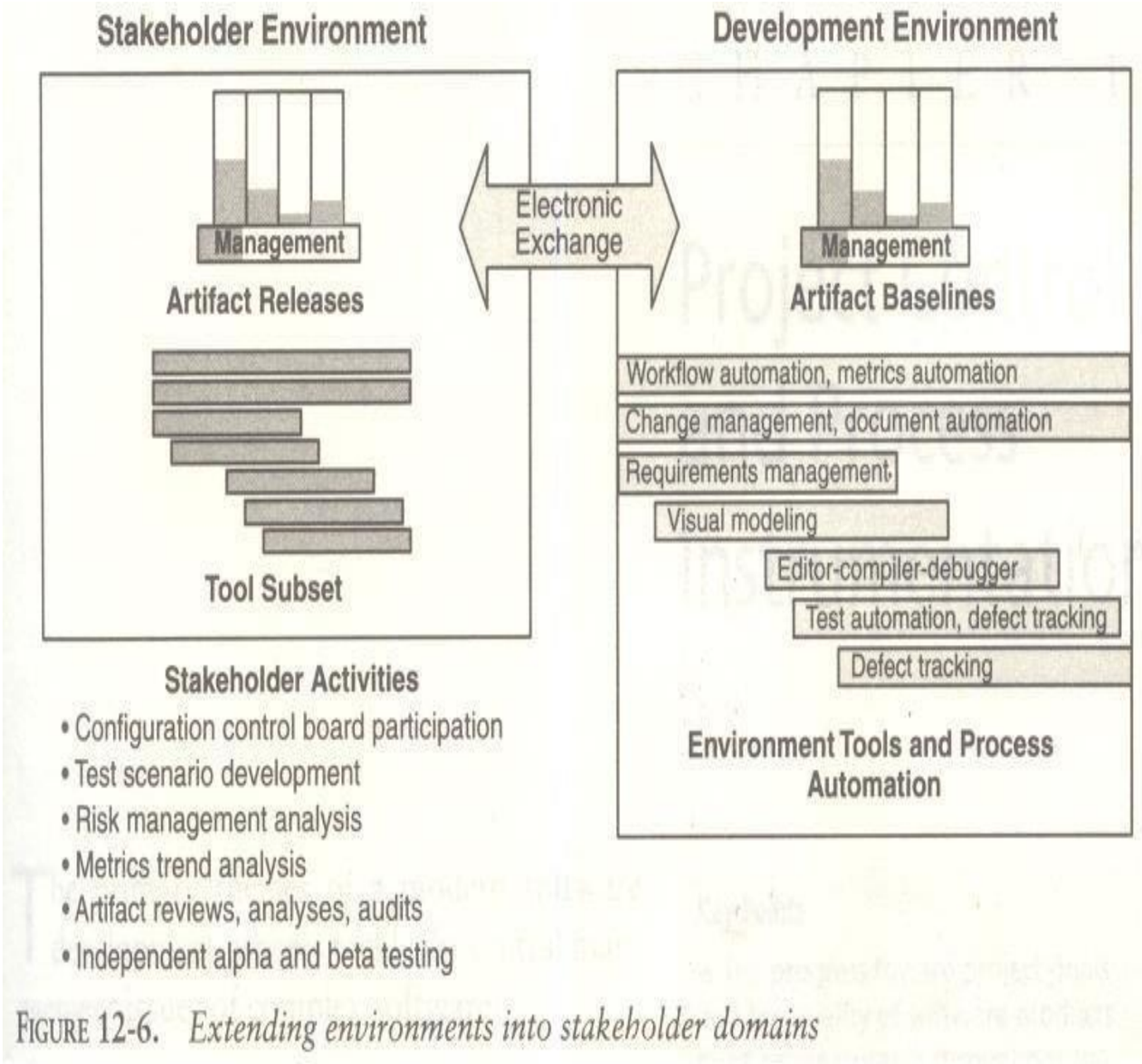


FIGURE 12-6. *Extending environments into stakeholder domains*

PROJECT CONTROL & PROCESS 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 inthe 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.

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 andproduce metrics from which the indicators can be derived.

Two types of indicators are:

(i) Management indicators.

(ii) Quality indicators.

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 thatindicate financial status are based on earned value system.
Quality Indicators
The quality indicators are based on the measurement of the changes occurred in software.

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 theprocess. The most useful metrics are extracted directly from the evolving artifacts.
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 Budgeted cost and expenditures Staffing and team dynamics	Change traffic and stability Breakage and modularity Rework and adaptability Mean time between failures (MTBF) and maturity

MANAGEMENT INDICATORS:

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 bemeasured 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

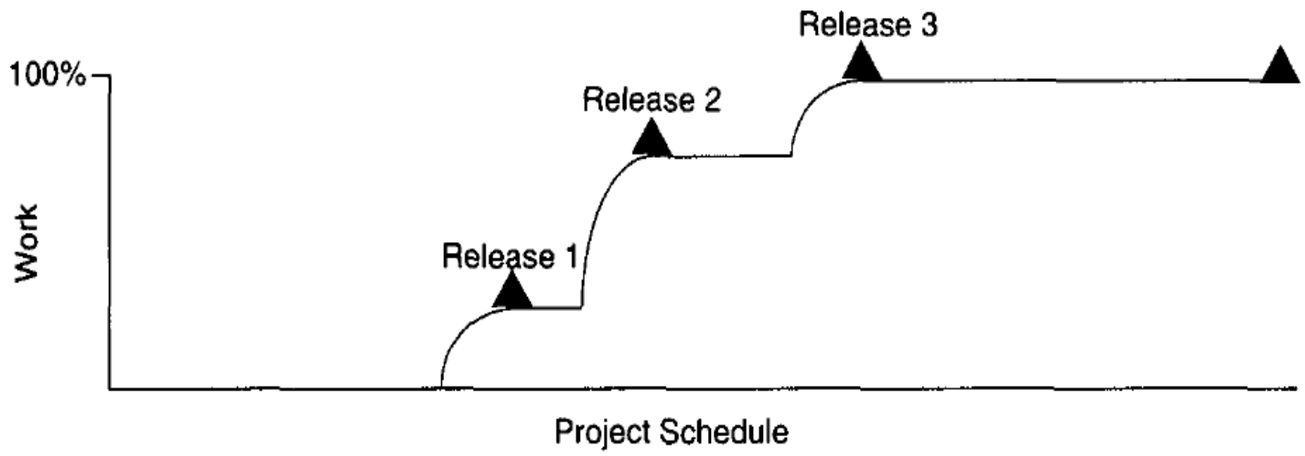


Fig: work and progress

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 anearned 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 centred 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.

Staffing and team dynamics

This metric measures 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.

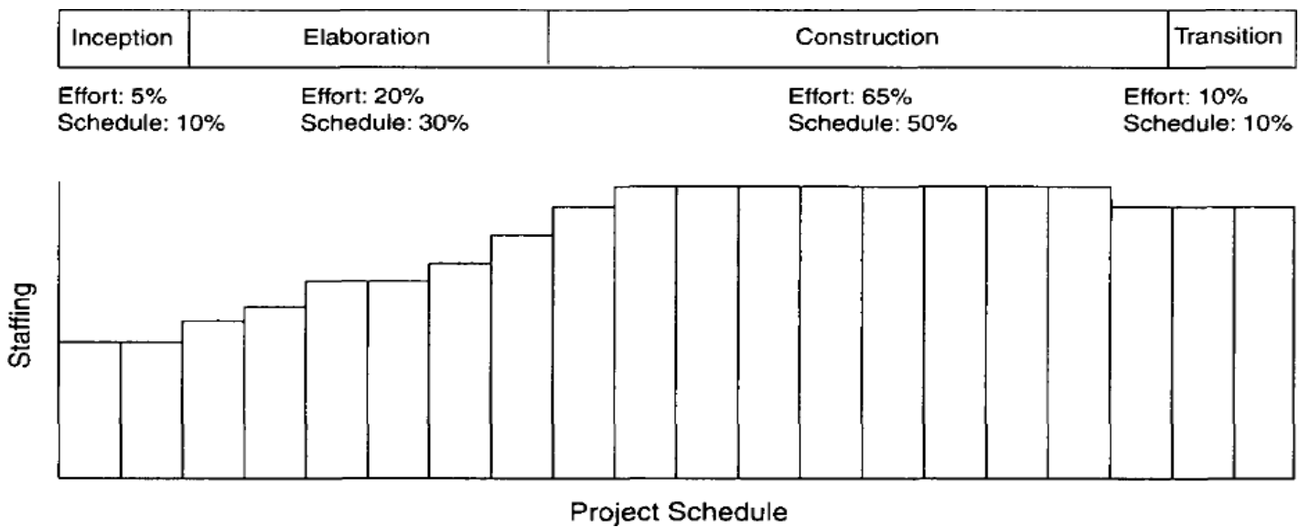


Fig: staffing and Team dynamics

QUALITY INDICATORS:

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

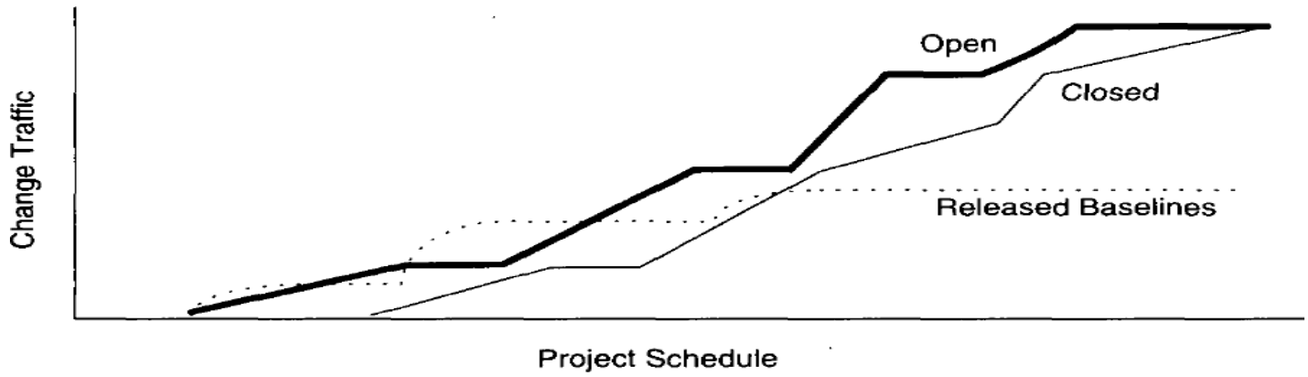


Fig: Change traffic and stability

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.

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 analyse, 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.

MTBF and Maturity:

This metric measures defect rather 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.

LIFE -CYCLE EXPECTATIONS:

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

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

They provide inside into the waste generated by the process. Scrap and rework metrics are a standardmeasurement 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 and the current trend provides tangible indicators for managementaction.

Table 13-3. the default pattern of life cycle 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

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:

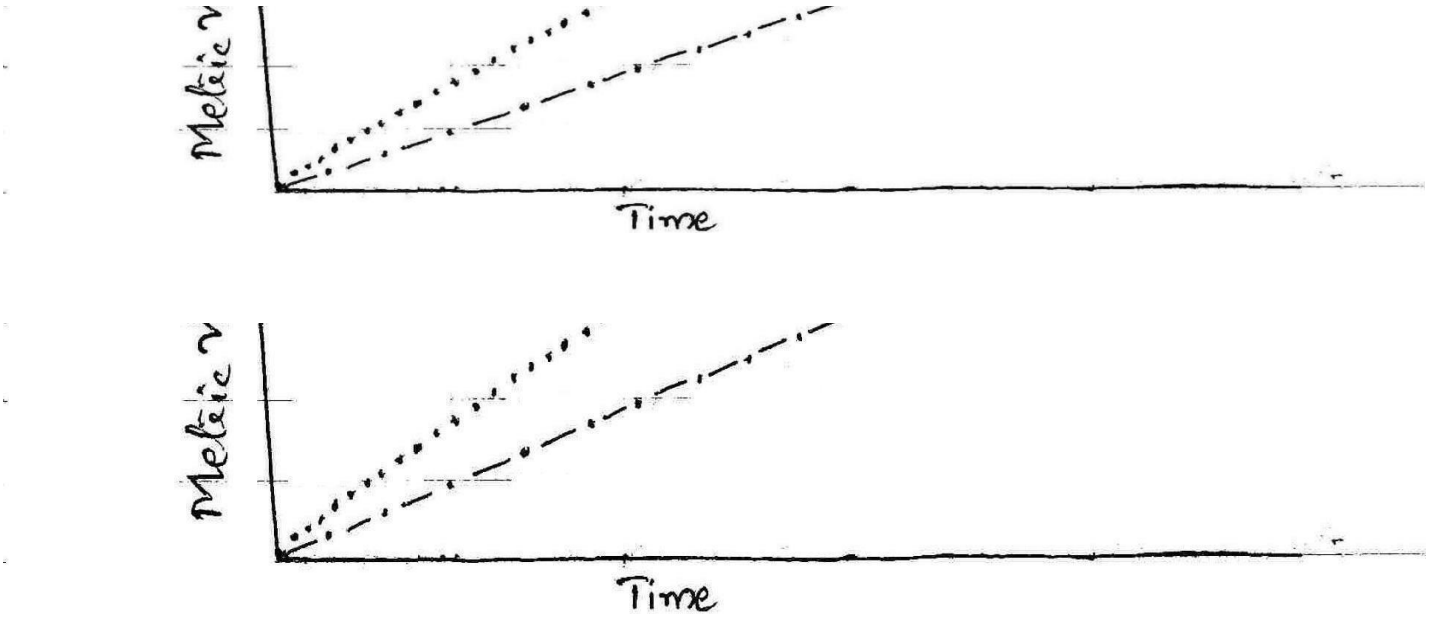
To implement a complete SPCP, the following are necessary.

- 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. A panel contains a number of graphical objects positioned in a particular geometric layout. A metric shown in a graphical object is labelled 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. 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.

IMP Questions

1. Define metric. Discuss seven core metrics for project control and process instrumentationwith suitable examples?
2. List out the three management indicators that can be used as core metrics on softwareprojects. Briefly specify meaning of each?
3. Explain the various characteristics of good software metric. Discuss the metrics Automationusing appropriate example?
4. Explain about the quality indicators that can be used as core metrics on software projects.
5. Explain Management Indicators with suitable example?
6. Define MTBF and Maturity. How these are related to each other?
7. Briefly explain about Quality Indicators?
8. Write short notes on Pragmatic software metrics?