RGPVNOTES.IN

Program : **B.Tech**

Subject Name: **Project Management**

Subject Code: **CS-604**

Semester: **6th**

Department of Computer Science and Engineering
Subject Notes
CS-604 (B) Project Management
Unit -3

Topics to be covered

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

1. Iterative Process Planning

1.1 Work Breakdown Structures

The development of a work breakdown structure is dependent on the project management style, organizational culture, customer preference, financial constraints and several other hard-to-define parameters .A WBS is simply a hierarchy of elements that decomposes the project plan into the discrete work tasks.

A WBS provides the following information structure:

1. A delineation of all significant work.
2. A clear task decomposition for assignment of responsibilities.
3. A framework for scheduling, budgeting, and expenditure tracking.

Two simple planning guidelines should be considered when a project plan is being initiated or assessed.

FIRST-LEVEL WBS ELEMENT	DEFAULT BUDGET
Management	10%
Environment	10%
Requirements	10%
Design	15%
Implementation	25%
Assessment	25%
Deployment	5%

Table 3.1 The First Guideline Prescribes a Default Allocation of Costs Among The First-Level WBS Elements

DOMAIN	INCEPTION	ELABORATION	CONSTRUCTION	TRANSITION
Effort	5%	20%	65%	10%
Schedule	10%	30%	50%	10%

Table 3.2 The Second Guideline Prescribes the Allocation of Effort and Schedule Across the Life-Cycle Phases

1.2 The Cost and Schedule Estimating Process: -

Forward-looking: 1. The software project manager develops a characterization of the overall size, process, environment, people, and quality required for the project

2. A macro-level estimate of the total effort and schedule is developed using a software cost estimation model

3. The software project manager partitions the estimate for the effort into a top-level WBS, also partitions the schedule into major milestone dates and partitions the effort into a staffing profile.

4. At this point, subproject managers are given the responsibility for decomposing each of the WBS elements into lower levels using their top-level allocation, staffing profile, and major milestone dates as constraints.

Backward-looking:

1. The lowest level WBS elements are elaborated into detailed tasks, for which budgets and schedules are estimated by the responsible WBS element manager.
2. Estimates are combined and integrated into higher level budgets and milestones.
3. Comparisons are made with the top-down budgets and schedule milestones. Gross differences are assessed and adjustments are made in order to converge on agreement between the top-down and the bottom-up estimates.

1.3. The Iteration Planning Process

Engineering Stage		Production Stage	
Inception	Elaboration	Construction	Transition
Feasibility	Architecture iterations	Usable iterations	Product releases

Table 3.3 The Iteration Planning Process

Engineering stage planning emphasis:

- Macro-level task estimation for production-stage artifacts
- Micro-level task estimation for engineering artifacts
- Stakeholder concurrence
- Coarse-grained variance analysis of actual vs. planned expenditures
- Tuning the top-down project-independent planning guidelines into project-specific planning guidelines.

Production stage planning emphasis:

- Micro-level task estimation for production-stage artifacts
- Macro-level task estimation for engineering artifacts
- Stakeholder concurrence
- Fine-grained variance analysis of actual vs. planned expenditures

2. Project Organizations and Responsibilities:

- **Organizations** engaged in software Line-of-Business need to support projects with the infrastructure necessary to use a common process.
- **Project** organizations need to allocate artifacts & responsibilities across project team to ensure a balance of 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 & probabilities.
- **Project teams** are motivated by the cost, Schedule & quality of specific deliverables

2.1 Line-Of-Business Organizations: The main features of default organization are as follows:

- Responsibility for process definition and maintenance is specific to a cohesive line of business, where process commonality makes sense. For example, the process for developing avionics software is different from the process used to develop office applications.
- Responsibility for process automation is an organizational role and is equal in importance to the process definition role. Projects achieve process commonality primarily through the environment support of common tools.
- Organizational roles may be fulfilled by a single individual or several different teams, depending on the scale of the organization. A 20-person software product company may require only a single

person to fulfill all the roles, while a 10,000-person telecommunications company may require hundreds of people to achieve an effective software organization.

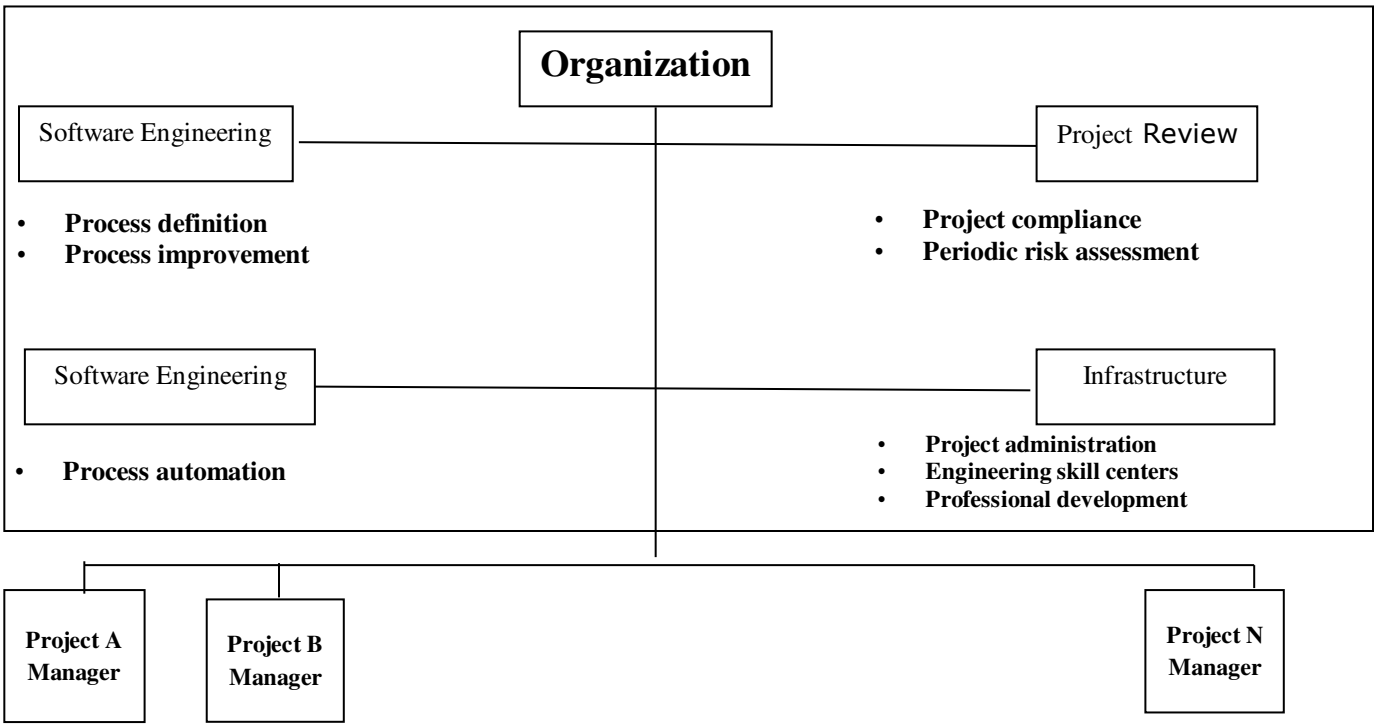


Fig 3.1 Default roles in a software Line-of-Business Organization

Software Engineering Process Authority (SEPA)

The SEPA facilities the exchange of information & process guidance both to & from project practitioners This role is accountable to General Manager for maintaining a current assessment of the Organization's process maturity & s plan for future improvement.

Project Review Authority (PRA)

The PRA is the single individual responsible for ensuring that a software project complies with all organizational & business un software policies, practices & standards. A software Project Manager is responsible for meeting the requirements of a contract or some other project compliance standard.

Software Engineering Environment Authority (SEEA)

The SEEA is responsible for automating the organization's process, maintaining the organization's standard environment, Training projects to use the environment & maintaining organization-wide reusable assets. The SEE A role is necessary to achieve a significant ROI for common process.

2.2 Infrastructure

An organization's infrastructure provides human resources support, project-independent research & development, & other capable software engineering assets.

Project Organizations

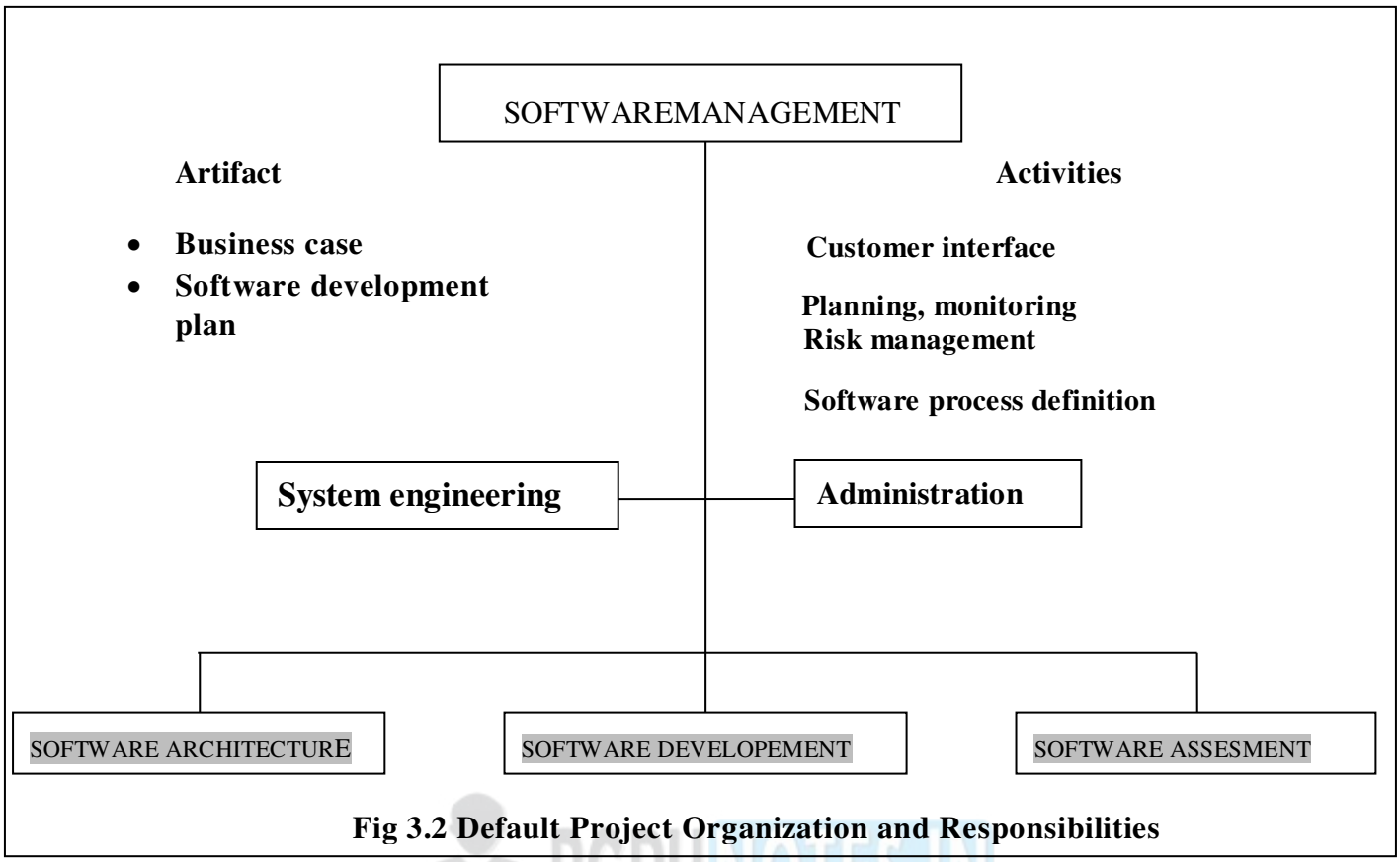


Fig 3.2 Default Project Organization and Responsibilities

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

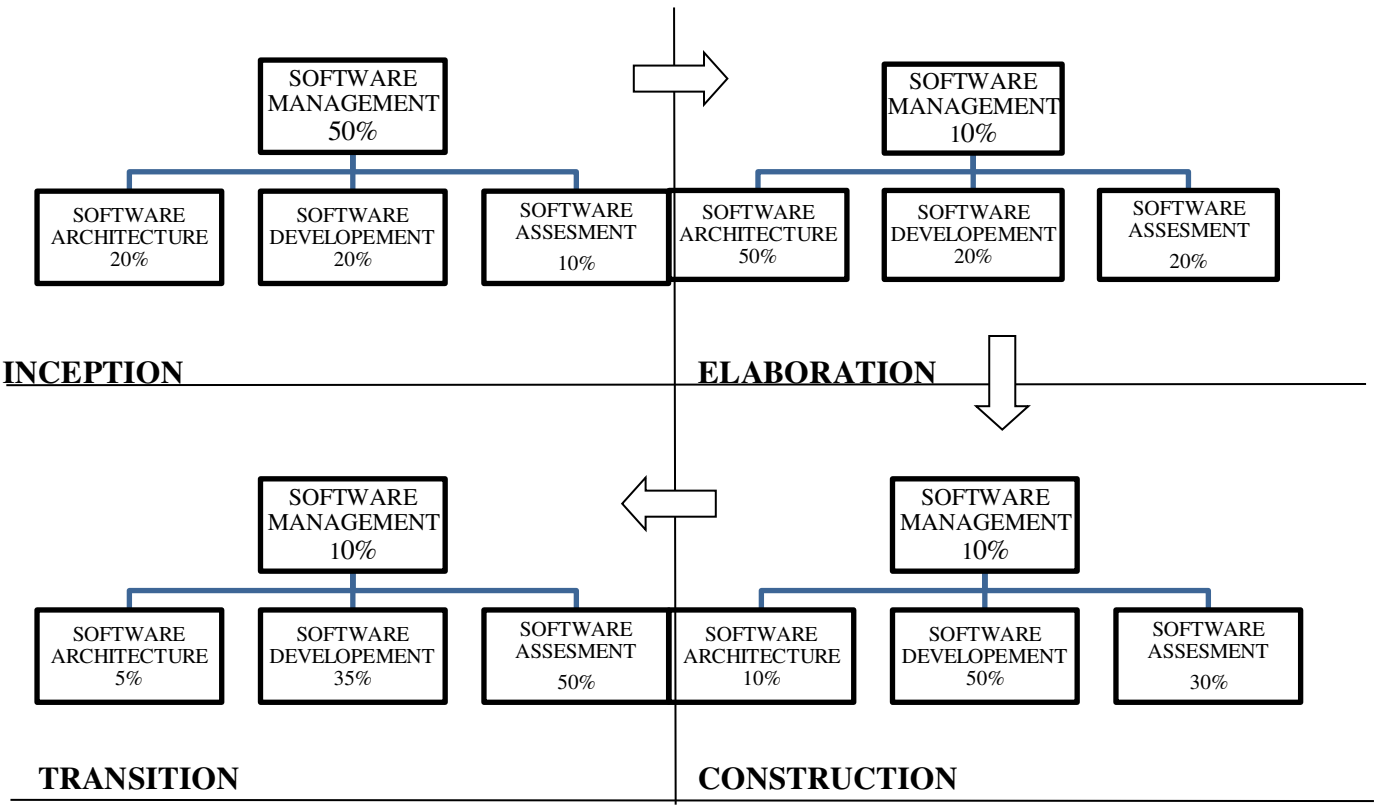


Fig 3.3 EVOLOTIONS OF ORGANIZATIONS

3. The Process Automation

Introductory Remarks:

The environment must be the first-class artifact of the process. Process automation & change management is crucial to an iterative process. If the change is expensive then the development organization will resist. Round-trip engineering & integrated environments promote change freedom & effective evolution often technical artifacts. Metric automation is crucial to effective project control. External stakeholders need access to environment resources to improve interaction with the development n team & add value to the process. The three levels of process which requires a certain degree of process automation for the corresponding process to be carried out efficiently.

1. Meta process (Line of business): The automation support for this level is called an infrastructure.

2. Macro process (Project): The automation support for a project's process is called an environment.

3. Micro process (Iteration): The automation support for generating artifacts is generally called a tool.

Automation Building blocks:

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

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

Table 3.4 Typical Automation and Tool Components that Supports Process Workflows

3.1 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 evaluate trade-offs during inception & elaboration phase of the life cycle. The Development environment should include a full use of development tools needed to support various Process 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 a modern iterative development process.

Round-Trip Engineering

Change Management

Software Change Orders (SCO)
Configuration baseline Configuration Control Board
Infrastructure
Organization Policy
Organization Environment

Stakeholder Environment.

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 iterative development. As the software industry moves into maintaining different information sets for the engineering artifacts, more automation 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 engineering artifacts.

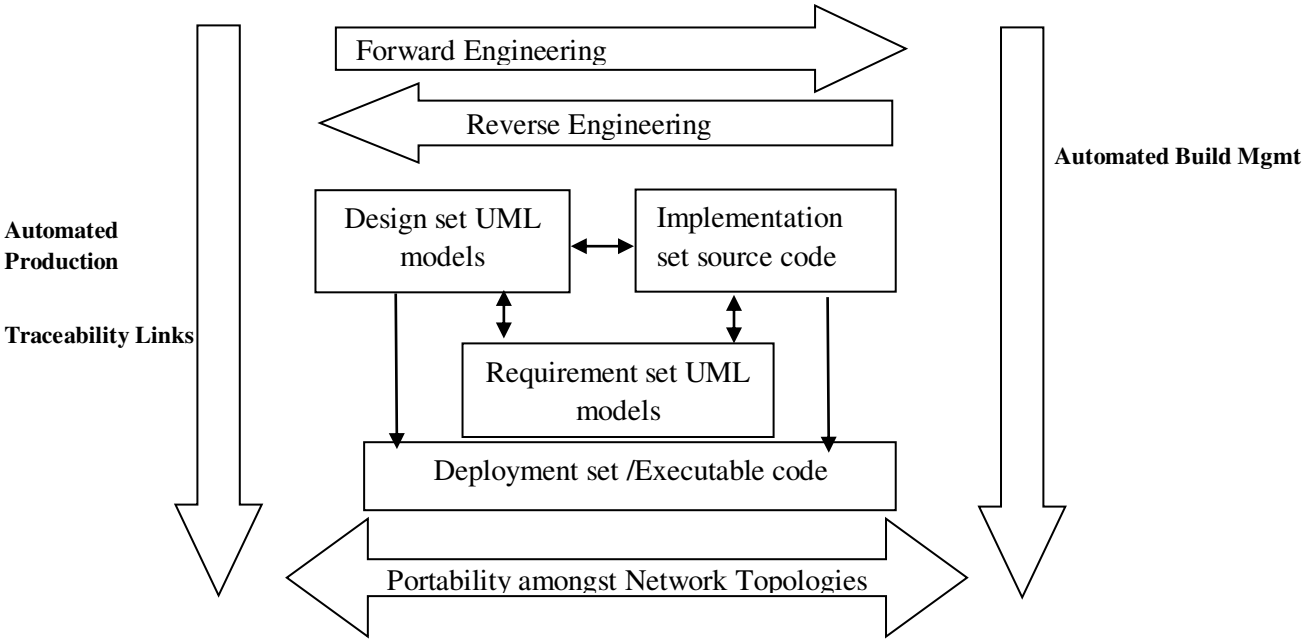


Fig 3.4 Round Trip Engineering

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 configuration baseline is called a software change orders (SCO).

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

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. There are generally two classes of baselines

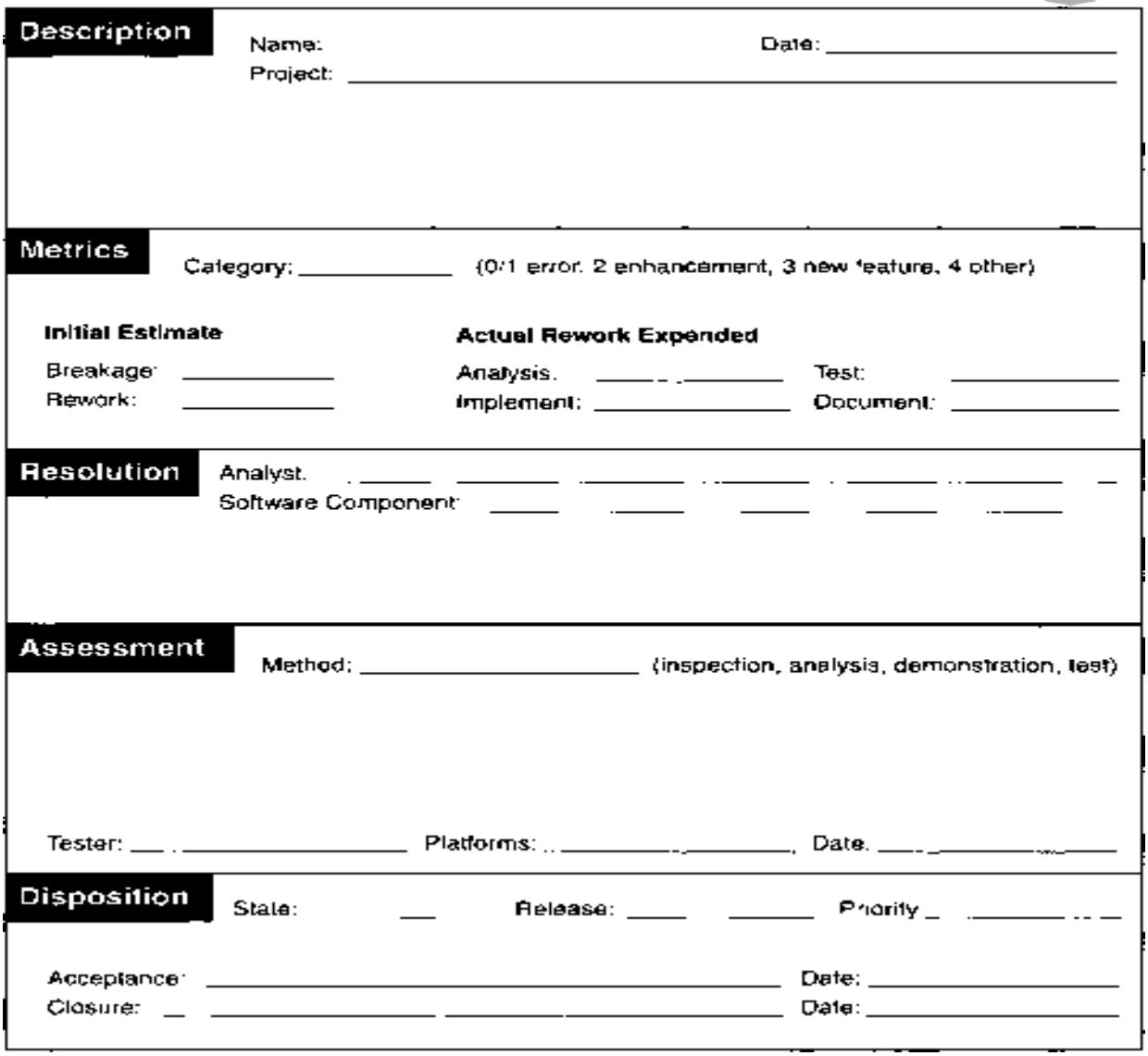
External Product Release

Internal 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 mustbe made 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 beworked around.

Type 2: A change that is an enhancement rather than a response to a defect

Type 3: A change that is associated by the update to the environment

Type 4: Changes that are not accommodated by the other categories.

III Configuration Control Board (CCB)

A CCB is a team of people that functions as the decision. Authority 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 delivery system.

nfrastructure

The organization infrastructure provides the organization's capital assets including two key artifacts - 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

a. Major milestones
b. Intermediate Artifacts
c. Engineering repository
d. Metrics
e. Roles & Responsibilities

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 stakeholder participating in the development process they might include

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

These stakeholder representatives also need to access to development resources so that they can contribute value to overall effort. These stakeholders will be access through on-line. An on-line environment accessible by the external stakeholders allows them to participate in the process a follow 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 & other overhead cost.

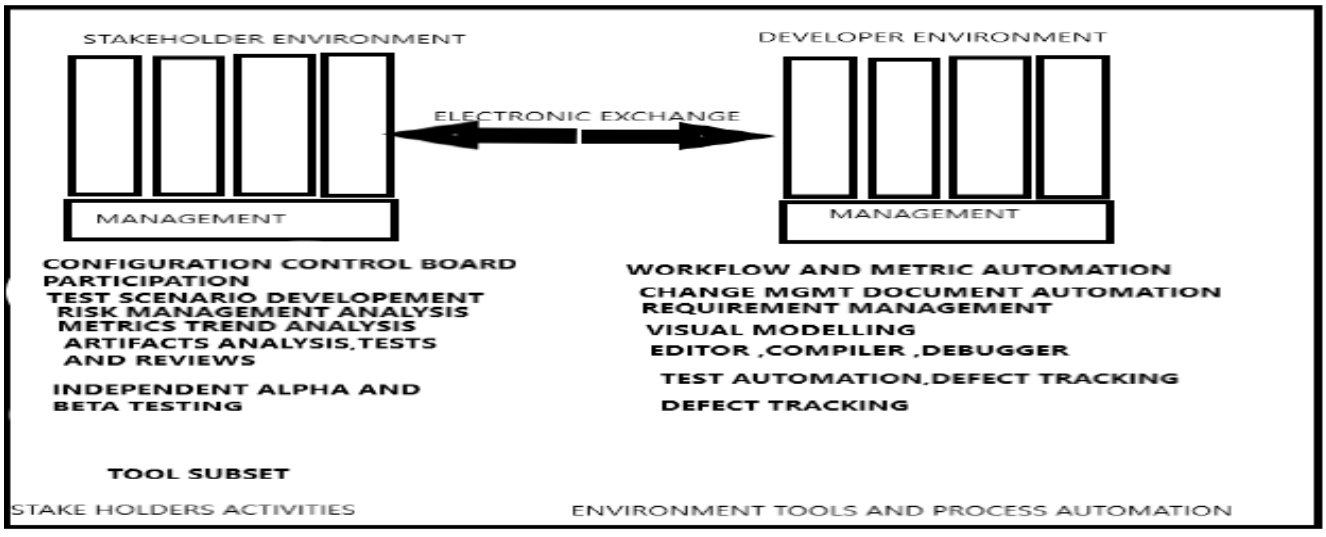


Fig 3.5 Extending environments into stakeholders domain

4. Project control and process instrumentation

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:

1. Software metrics are needed for calculating the cost and schedule of a software product with great accuracy.
2. Software metrics are required for making an accurate estimation of the progress.
3. 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 and produce 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 that indicate financial status are based on earned value system.

Quality Indicators

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

5. SEVEN CORE METRICS OF SOFTWARE PROJECT

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

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

The seven-core metrics can be used in numerous ways to help manage projects and organizations. In an iterative development project or an organization structured around a software line of business, the historical values of previous iterations and projects provide precedent data for planning subsequent iterations and projects. Consequently, once metrics collection is ingrained, a project or organization can improve its ability to predict the cost, schedule, or quality performance of future work activities.

The seven-core metrics are based on common sense and field experience with both successful and unsuccessful metrics programs. Their attributes include the following:

- They are simple, objective, easy to collect, easy to interpret, and hard to misinterpret.
- Collection can be automated and nonintrusive.

• They provide for consistent assessments throughout the life cycle and are derived from the evolving product baselines rather than from a subjective assessment.
• They are useful to both management and engineering personnel for communicating progress and quality in a consistent format.
• Their fidelity improves across the life cycle.

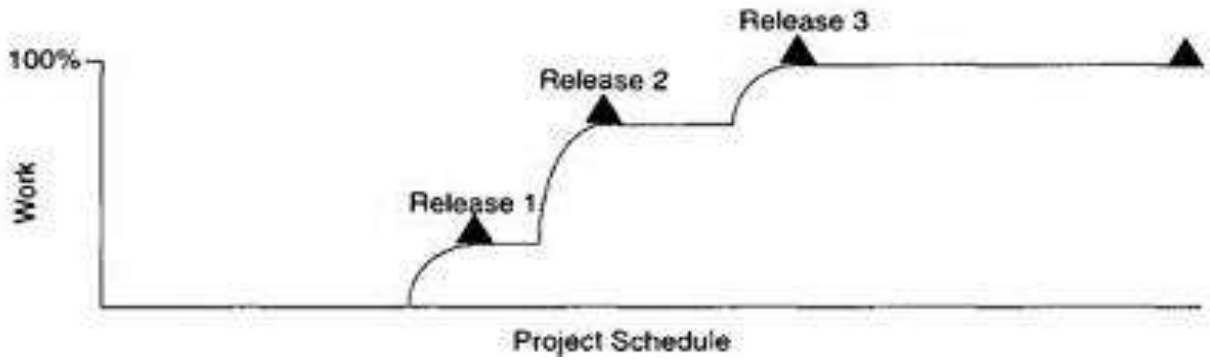


FIG 3.6 Expected Progress for typical project with three major releases

6. 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 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.

Budgeted Cost and Expenditures: - releases, by term, by components, by subsystems, etc

This metric measure 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 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 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 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 centred in 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

These 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.

QUALITY INDICATORS:

The four quality indicators are based primarily on the measurement of software change across evolving baselines of engineering data (such as design models and source code).

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

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 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.

MTBF and Maturity:

This metric measure defect rate 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 error 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 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.

7. **Life Cycle Expectations**

There is no mathematical or formal derivation for using seven core metrics properly. However, there were specific 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 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 and the current trend provides tangible indicators for management action.

Metric	Elaboration	Construction	Transition	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

Table 3.5 The default pattern of life cycle evolution

8. Process Discriminate

In tailoring the management process to a specific domain or projects, there are two discriminating factors: -1. Technical complexity 2. Management complexity. The formality of reviews, the quality of artifacts, the priorities of concerns and numerous other process instantiation parameters are governed by a point project occupies in these two dimensions. A well-defined software process is critical for success in software projects. Software process tailoring refers to the activity of tuning a standardized process to meet the needs of a specific project. Tailoring is necessary because each project is unique; not every process, tool, technique, input, or output identified is required on every project. Tailoring should address the competing constraints of scope, schedule, cost, resources, quality, and risk. A process framework must be configured to the specific characteristic of the project. The Process discriminants are organized around six process parameters –scale, stake holder cohesion, process flexibility, process maturity, architectural risk ad 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.

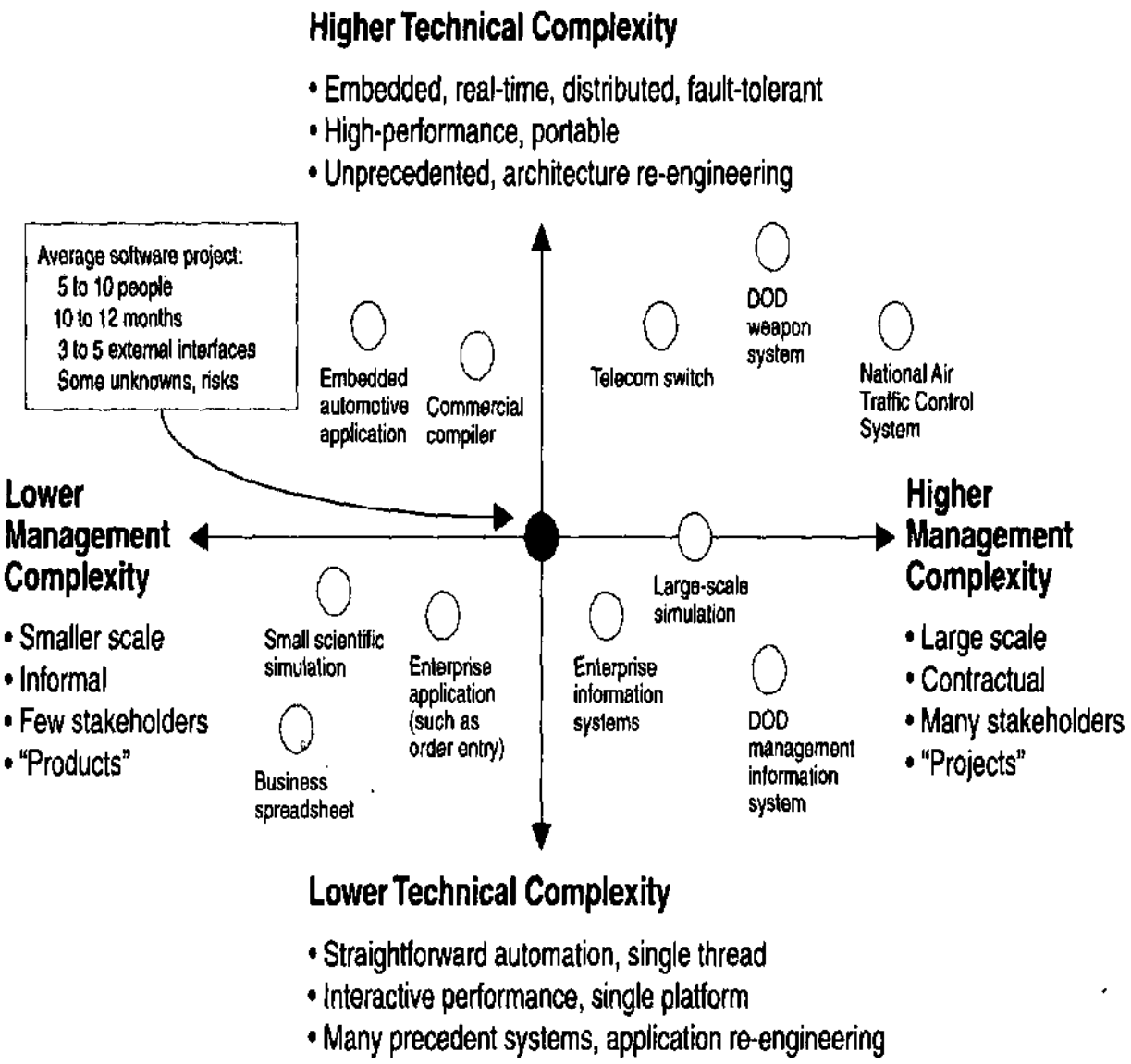


Fig 3.7 Priorities for Tailoring a Process Framework

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.

A process framework is not a project specific process implementation with a well defined recipe of success. Judgement must be injected and the methods, techniques, culture, formality, organization must be tailored to the specific domain to achieve a process implementation that can succeed.

Process Primitive	Smaller Team	Larger Team
Life cycle phases	Weak boundaries between phases	Well defined phase synchronize progress among concurrent activities
Artifacts	Focus on technical artifacts, few discrete baselines, very few mgmt. artifacts required.	Change management of technical artifacts which May result in numerous baselines, management artifacts important
Workflow effort allocations	More need for generation, people who perform roles in multiple workflows	Higher percentage of specialist. more people and team focus on specific workflow.
Checkpoints	Many informal events for maintaining technical consistency, no schedule disruption.	A few formal events synchronize among teams which can take days.
Management discipline	Informal planning disruption managed by individual	Formal organisations, formal planning, project control,
Automation discipline	More ad hoc environments managed by individual.	Infrastructure to ensure consistent, up to date environment available access all teams

Table 3.6 Process discriminators that results from differences in project size

Process Primitive	Few stake holders	Multiple stakeholders, Adversarial Relationships
Life cycle phases	Weak boundaries between phases	Well defined phase synchronize progress among concurrent activities
Artifacts	Fewer and less detailed management artifacts required.	Management artifacts paramount, especially the business case.
Workflow effort allocations	Lesser overhead assessment	High assessment overhead to ensure stakeholder concurrence
Checkpoints	Many informal events	3 to 4 formal events.
Management discipline	Informal planning, project control and organisation.	formal planning, project control,
Automation discipline	Insignificant.	Online stake holder environment necessary.

Table 3.7 Process discriminators that results from differences in stakeholder cohesion

Process Primitive	Flexible process	Inflexible process
Life cycle phases	Tolerant to cavalier phase commitments	More credible basis required for inception phase commitments
Artifacts	Changeable business case vision	Carefully controlled changes to business case vision
Workflow effort allocations	Insignificant	Increased levels of management and assessment workflows
Checkpoints	Many informal events for maintaining technical consistency	3 to 4 formal events.
Management discipline	Insignificant	More fidelity required for planning and project cp
Automation discipline	Insignificant.	Insignificant

Table 3.8 Process discriminators that results from differences Process flexibility

Process Primitive	Mature level 3 or 4 organization	Level 1 Organization
Life cycle phases	Well established criteria for phase transitions	Insignificant
Artifacts	Well established format, content and production methods	Free form
Workflow effort allocations	Insignificant	No basis
Checkpoints	Well defined	Insignificant.
Management discipline	Predictable planning, objective status planning	Informal planning and project control.
Automation discipline	Requires high level s of automation for round trip engineering, change management	Little automation

Table 3.9 Process discriminators that results from differences in project maturity

Process Primitive	Complete architecture feasibility demonstration	No architecture feasibility demonstration
Life cycle phases	More inception and elaboration phase iterations	Fewer early iterations/more constructions iterations
Artifacts	Earlier breadth and depth across technical artifacts	Insignificant
Workflow effort allocations	Higher level of design efforts/lower level of implementation and assessment	Higher levels of implementation and assessment to deal.
Checkpoints	More emphasis on executable demonstrations	More emphasis on briefings, documentation and simulation.
Management discipline	Insignificant	Insignificant
Automation discipline	More environment resources required earlier in the life cycle	Less environment demand early in the life cycle.

Table 3.10 Process discriminators that results from differences in architectural risk

Process Primitive	Experienced Team	Inexperienced Team
Life cycle phases	Shorter engineering stage	Longer engineering stage
Artifacts	Less scrape and rework in requirement and design set	More scrap and rework in requirement and design sets
Workflow effort allocations	Lower level of requirement and design	Higher levels of requirements and design
Checkpoints	Insignificant	Insignificant
Management discipline	Less emphasis on risk management/less frequent status assessments needed	More frequent status assessments required
Automation discipline	Insignificant	Insignificant

Table 3.11 Process discriminators that results from differences in domain experience

RGPVNOTES.IN

We hope you find these notes useful.

You can get previous year question papers at https://qp.rgpvnotes.in .

If you have any queries or you want to submit your study notes please write us at rgpvnotes.in@gmail.com

LIKE & FOLLOW US ON FACEBOOK
facebook.com/rgpvnotes.in