



Program : **B.Tech**

Subject Name: **Project Management**

Subject Code: **CS-604**

Semester: **6th**



LIKE & FOLLOW US ON FACEBOOK

facebook.com/rgpvnotes.in

Department of Computer Science and Engineering
Subject Notes
CS-604 (B) Project Management
Unit -2

Topics to be covered

Software Management Process

Framework: Life cycle phases- inception, elaboration, construction and training phase. Artifacts of the process- the artifact sets, management artifacts, engineering artifacts, pragmatics artifacts. Model based software architectures. Workflows of the process. Checkpoints of the process.

Engineering and Production Stages:

To achieve economies of scale and higher return on investment, we must move toward a software manufacturing process which is determined by technological improvements in process automation and component based development.

There are two stages in the software development process:



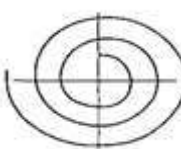
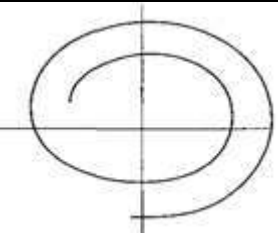
- 1) **The Engineering Stage:** Less predictable but smaller teams doing design and production activities. This stage is decomposed into two distinct Phases inception and elaboration.
- 2) **The Production Stage:** More predictable but larger teams doing construction, test, and deployment activities. This stage is also decomposed into two distinct Phases construction and transition.

Table 2.1 the Two Stages of the Life Cycle: Engineering and Production

S no.	Life Cycle Aspects	Engineering Stage Emphasis	Production Stage Emphasis
1	Risk Reduction	Schedule, Technical, Feasibility	Cost
2	Products	Architecture baseline	Product release baseline
3	Activities	Analysis, Design, Planning	Implementation, Testing
4	Assessment	Demonstration, Inspection, Analysis	Testing
5	Economics	Resolving diseconomies of scale	Exploiting Economies of scale
6	Management	Planning	Operations

These four phases of lifecycle process are loosely mapped to the conceptual framework of the spiral model is as shown in the following table:

Table 2.2 Four phases of lifecycle

Engineering Stage		Production Stage	
Inception	Elaboration	Construction	Transition
			
Idea	Architecture	Beta Releases	Products



- In the above figure the size of the spiral corresponds to the inactivity of the project with respect to the breadth and depth of the artifacts that have been developed.
- This inertia manifests itself in maintaining artifact consistency, regression testing, documentation, quality analyses, and configuration control.
- Increased inertia may have little, or at least very straightforward, impact on changing any given discrete component or activity.
- However, the reaction time for accommodating major architectural changes, major requirements changes, major planning shifts, or major organizational perturbations clearly increases in subsequent phases.

Inception Phase:

The main goal of this phase is to achieve agreement among stakeholders on the life-cycle objectives for the project.

Primary Objectives:

- 1) Establishing the project's scope and boundary conditions
- 2) Distinguishing the critical use cases of the system and the primary scenarios of operation
- 3) Demonstrating at least one candidate architecture against some of the primary scenarios
- 4) Estimating cost and schedule for the entire project
- 5) Estimating potential risks

Essential Activities:

- 1) Formulating the scope of the project
- 2) Synthesizing the architecture
- 3) Planning and preparing a business case

Elaboration Phase:

- It is the most critical phase among the four phases.
- Depending upon the scope, size, risk, and freshness of the project, an executable architecture prototype is built in one or more iterations.
- At most of the time the process may accommodate changes, the elaboration phase activities must ensure that the architecture, requirements, and plans are stable. And also the cost and schedule for the completion of the development can be predicted within an acceptable range.

Primary Objectives

- 1) Base lining the architecture as rapidly as practical
- 2) Base lining the vision
- 3) Base lining a high-reliability plan for the construction phase
- 4) Demonstrating that the baseline architecture will support the vision at a reasonable cost in a reasonable time.

Essential Activities

- 1) Elaborating the vision
- 2) Elaborating the process and infrastructure
- 3) Elaborating the architecture and selecting components

Construction Phase

During this phase all the remaining components and application features are integrated into the application and all features are thoroughly tested. Newly developed software is integrated where ever required. If it is a big project then parallel construction increments are generated.

Primary Objectives

- 1) Minimizing development costs
- 2) Achieving adequate quality as rapidly as practical
- 3) Achieving useful version (alpha, beta, and other releases) as rapidly as practical

Essential Activities

- 1) Resource management, control, and process optimization
- 2) Complete component development and testing evaluation criteria
- 3) Assessment of product release criteria of the vision

Transition Phase

Whenever a project is grown-up completely and to be deployed in the end-user domain this phase is called transition phase. It includes the following activities:

- 1) Beta testing to validate the new system against user expectations
- 2) Beta testing and parallel operation relative to a legacy system it is replacing
- 3) Conversion of operational databases
- 4) Training of users and maintainers

Primary Objectives

- 1) Achieving user self-supportability
- 2) Achieving stakeholder concurrence
- 3) Achieving final product baseline as rapidly and cost-effectively as practical

Essential Activities

- 1) Synchronization and integration of concurrent construction increments into consistent deployment baselines
- 2) Deployment-specific engineering
- 3) Assessment of deployment baselines against the complete vision and acceptance criteria in the requirement set.

Artifacts of the Process:

- Conventional s/w projects focused on the sequential development of s/w artifacts:
- Build the requirements
- Construct a design model traceable to the requirements
- Compile and test the implementation for deployment.
- This process can work for small-scale, purely custom developments in which the design representation, implementation representation and deployment representation are closely aligned.
- This approach is doesn't work for most of today's s/w systems why because of having complexity and are composed of numerous components some are custom, some reused, some commercial products.

The Artifacts Sets

In order to manage the development of a complete software system, we need to gather distinct collections of information and is organized into artifact sets.

- Set represents a complete aspect of the system where as artifact represents interrelated information that is developed and reviewed as a single entity.
- The artifacts of the process are organized into five sets:
1) Management 2) Requirements 3) Design 4) Implementation 5) Deployment

Here the management artifacts capture the information that is necessary to synchronize stakeholder expectations. Whereas the remaining four artifacts are captured rigorous notations that support automated analysis and browsing

Table 2.3 The Artifacts Sets

Requirements Set	Design Set	Implementation Set	Deployment Set
<ol style="list-style-type: none"> 1. Vision document 2. Requirements model(s) 	<ol style="list-style-type: none"> 1. Design Model(s) 2. Test model 3. Software architecture description 	<ol style="list-style-type: none"> 1. Source code baseline 2. Associated Compile-time files 3. Component executables 	<ol style="list-style-type: none"> 1. Integrated product executable baseline 2. Associated run time files 3. User manual

Management Set	
Planning Artifacts	Operational Artifacts
<ol style="list-style-type: none"> 1. Work Breakdown structure 2. Business case 3. Release Specifications 4. Software Development Plan 	<ol style="list-style-type: none"> 1. Release descriptions 2. Status assessments 3. Software change order database 4. Deployment documents and Environment

The Management Set

It captures the artifacts associated with process planning and execution. These artifacts use ad hoc notation including text, graphics, or whatever representation is required to capture the “contracts” among,

- **project personnel:** project manager, architects, developers, testers, marketers, administrators
- **stakeholders:** Funding authority, user, s/w project manager, organization manager, regulatory agency & between project personnel and stakeholders

Management artifacts are evaluated, assessed, and measured through a combination of

- 1) Relevant stakeholder review.
- 2) Analysis of changes between the current version of the artifact and previous versions.
- 3) Major milestone demonstrations of the balance among all artifacts.

The Engineering Sets:

1) Requirement Set:

- The requirement set is the primary engineering context for evaluating the other three engineering artifact sets and is the basis for test cases.

- **Requirement artifacts are evaluated, assessed, and measured through a combination of**

- 1) Analysis of consistency with the release specifications of the mgmt set.
- 2) Analysis of consistency between the vision and the requirement models.
- 3) Mapping against the design, implementation, and deployment sets to evaluate the consistency and completeness and the semantic balance between information in the different sets.
- 4) Analysis of changes between the current version of the artifacts and previous versions.
- 5) Subjective review of other dimensions of quality.

2) Design Set:

- UML notations are used to engineer the design models for the solution.
- It contains various levels of abstraction and enough structural and behavioral information to determine a bill of materials.
- Design model information can be clearly and, in many cases, automatically translated into a subset of the implementation and deployment set artifacts.

The design set is evaluated, assessed, and measured through a combination of

- 1) Analysis of the internal consistency and quality of the design model.
- 2) Analysis of consistency with the requirements models.
- 3) Translation into implementation and deployment sets and notations to evaluate the Consistency and completeness and semantic balance between information in the sets.
- 4) Analysis of changes between the current version of the design model and previous versions.
- 5) Subjective review of other dimensions of quality.

3) Implementation set:

The implementation set include source code that represents the tangible implementations of components and any executables necessary for stand-alone testing of components.

- Executables are the primitive parts that are needed to construct the end product, including custom components, APIs of commercial components.
- Implementation set artifacts can also be translated into a subset of the deployment set.

Implementation sets are human-readable formats that are evaluated, assessed, and measured through a combination of:

- 1) Analysis of consistency with design models
- 2) Translation into deployment set notations to evaluate consistency and completeness among artifact sets.
- 3) Execution of stand-alone component test cases that automatically compare expected results with actual results.
- 4) Analysis of changes b/w the current version of the implementation set and previous versions.
- 5) Subjective review of other dimensions of quality.

4) Deployment Set:

It includes user deliverables and machine language notations, executable software, and the build scripts, installation scripts, and executable target-specific data necessary to use the product in its target environment.

Deployment sets are evaluated, assessed, and measured through a combination of:

- 1) Testing against the usage scenarios and quality attributes defined in the requirements set to evaluate the consistency and completeness and the semantic balance between information in the two sets.
- 2) Testing the partitioning, replication, and allocation strategies in mapping components of the implementation set to physical resources of the deployment system.
- 3) Testing against the defined usage scenarios in the user manual such as installation, user-oriented dynamic reconfiguration, mainstream usage, and anomaly management.
- 4) Analysis of changes b/w the current version of the deployment set and previous versions.
- 5) Subjective review of other dimensions of quality.

Each artifact set uses different notations to capture the relevant artifact.

- 1) **Management set notations** (ad hoc text, graphics, use case notation) capture the plans, process, objectives, and acceptance criteria.
- 2) **Requirement notation** (structured text and UML models) capture the engineering context and the operational concept.
- 3) **Implementation notations** (software languages) capture the building blocks of the solution in human-readable formats.
- 4) **Deployment notations** (executables and data files) capture the solution in machine-readable formats.

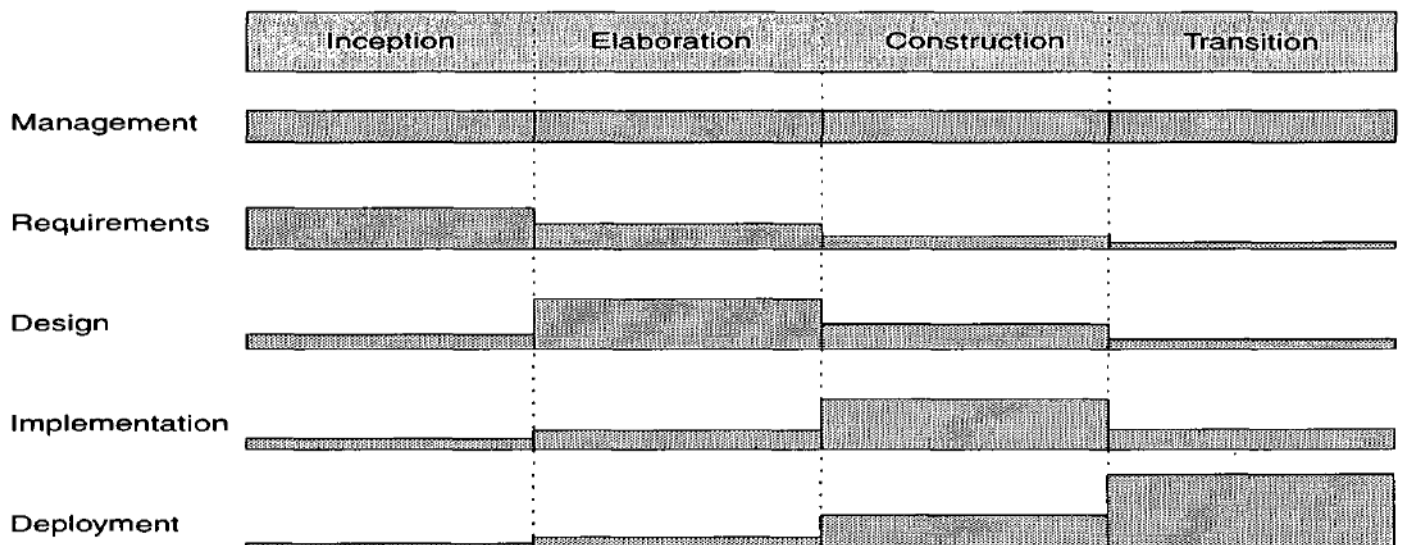


Figure 2.1 Different notations to capture the relevant artifact

Implementation Set versus Deployment Set

- A. The structure of the information delivered to the user (testing people) is very different from the structure of the source code implementation.
- B. Engineering decisions that have impact on the quality of the deployment set but are relatively incomprehensible in the design and implementation sets include:

- 1) Dynamically reconfigurable parameters such as buffer sizes, color palettes, number of servers, number of simultaneous clients, data files, run-time parameters.
- 2) Effects of compiler/link optimizations such as space optimization versus speed optimization.
- 3) Performance under certain allocation strategies such as centralized versus distributed, primary and shadow threads, dynamic load balancing.
- 4) Virtual machine constraints such as file descriptors, garbage collection, heap size, maximum record size, disk file rotations.
- 5) Process-level concurrency issues such as deadlock and race condition.
- 6) Platform-specific differences in performance or behavior.

Management Artifacts

The management set includes several artifacts that capture intermediate results and ancillary information necessary to document the product/process legacy, maintain the product, improve the product, and improve the process.

Business Case

The business case artifact provides all the information necessary to determine whether the project is worth investing in. It details the expected revenue, expected cost, technical and management plans, and backup data necessary to demonstrate the risks and realism of the plans. The main purpose is to transform the vision into economic terms so that an organization can make an accurate ROI assessment. The financial forecasts are evolutionary, updated with more accurate forecasts as the life cycle progresses.

Software Development Plan

The software development plan (SDP) elaborates the process framework into a fully detailed plan. Two indications of a useful SDP are periodic updating (it is not stagnant shelf ware) and understanding and acceptance by managers and practitioners alike.

Engineering Artifacts

Most of the engineering artifacts are captured in rigorous engineering notations such as UML, programming languages, or executable machine codes. Three engineering artifacts are explicitly intended for more general review, and they deserve further elaboration.

Vision Document

The vision document provides a complete vision for the software system under development and supports the contract between the funding authority and the development organization. A project vision is meant to be changeable as understanding evolves of the requirements, architecture, plans, and technology. A good vision document should change slowly.

Pragmatic Artifacts

- People want to review information but don't understand the language of the artifact. Many interested reviewers of a particular artifact will resist having to learn the engineering language in which the artifact is written. It is not uncommon to find people (such as veteran software managers, veteran quality assurance specialists, or an auditing authority from a regulatory agency) who react as follows: "I'm not going to learn UML, but I want to review the design of this software, so give me a separate description such as some flowcharts and text that I can understand."
- People want to review the information but don't have access to the tools. It is not very common for the development organization to be fully tooled; it is extremely rare that the/other stakeholders have any Capability to review the engineering artifacts on-line. Consequently, organizations are forced to exchange paper documents. Standardized formats (such as UML, spreadsheets, Visual Basic, C++, and Ada 95), visualization tools, and the Web are rapidly making it economically feasible for all stakeholders to exchange information electronically
- Human-readable engineering artifacts should use rigorous notations that are complete, consistent, and used in a self-documenting manner. Properly spelled English words should be used for all identifiers and descriptions. Acronyms and abbreviations should be used only where they are well accepted jargon in the context of the component's usage. Readability should be emphasized and the use of proper English words should be required in all engineering artifacts. This practice enables understandable representations, browse able formats (paperless review), more-rigorous notations, and reduced error rates.
- Useful documentation is self-defining: It is documentation that gets used.
- Paper is tangible; electronic artifacts are too easy to change. On-line and Web-based artifacts can be changed easily and are viewed with more skepticism because of their inherent volatility.

Model-Based Software Architectures

- Software architecture is the central design problem of a complex software system in the same way architecture is the software system design.
- The ultimate goal of the engineering stage is to converge on a stable architecture baseline.
- Architecture is not a paper document. It is a collection of information across all the engineering sets.
- Architectures are described by extracting the essential information from the design models.
- A model is a relatively independent abstraction of a system.
- A view is a subset of a model that abstracts a specific, relevant perspective.

Architecture: Management Perspective

The most critical and technical product of a software project is its architecture

- If a software development team is to be successful, the inter project communications, as captured in software architecture, must be accurate and precise.

From the management point of view, three different aspects of architecture

1. An architecture (the intangible design concept) is the design of software system it includes all engineering necessary to specify a complete bill of materials. Significant make or buy decisions are resolved, and all custom components are elaborated so that individual component costs and construction/assembly costs can be determined with confidence.
2. An architecture baseline (the tangible artifacts) is a slice of information across the engineering artifact sets sufficient to satisfy all stakeholders that the vision (function and quality) can be achieved within the parameters of the business case (cost, profit, time, technology, and people).
3. An architectural description is an organized subset of information extracted from the design set model's. It explains how the intangible concept is realized in the tangible artifacts.

The number of views and level of detail in each view can vary widely. For example the architecture of the software architecture of a small development tool.

There is a close relationship between software architecture and the modern software development Process because of the following reasons:

1. A stable software architecture is nothing but a project milestone where critical make/buy decisions should have been resolved. The life-cycle represents a transition from the engineering stage of a project to the production stage.
2. Architecture representation provide a basis for balancing the trade-offs between the problem space (requirements and constraints) and the solution space (the operational product).
3. The architecture and process encapsulate many of the important communications among individuals, teams, organizations, and stakeholders.
4. Poor architecture and immature process are often given as reasons for project failure.
5. In order to proper planning, a mature process, understanding the primary requirements and demonstrable architecture are important fundamentals.
6. Architecture development and process definition are the intellectual steps that map the problem to a solution without violating the constraints; they require human innovation and cannot be automated.

Architecture: Technical Perspective

Software architecture includes the structure of software systems, their behavior, and the patterns that guide these elements, their collaborations, and their composition. An architecture framework is defined in terms of views is the abstraction of the UML models in the design set. Where as architecture view is an abstraction of the design model, include full breadth and Depth of information.

Most real-world systems require four types of views:

- 1) Design: describes architecturally significant structures and functions of the design model.
- 2) Process: describes concurrency and control thread relationships among the design, component, and deployment views.
- 3) Component: describes the structure of the implementation set.
- 4) Deployment: describes the structure of the deployment set.

Workflows of the Process:

The term **workflow** means a thread of cohesive and mostly sequential activities. In most of the cases a process is a sequence of activities why because of easy to understand, represent, plan and conduct. But the simplistic activity sequences are not realistic why because it includes many teams, making progress on many artifacts that must be synchronized, cross-checked, homogenized, merged and integrated. In order to manage complex software's the workflow of the software process is to be changed that is distributed. Modern software process avoids the life-cycle phases like inception, elaboration, construction and transition. It tells only the state of the project rather than a sequence of activities in each phase. The activities of the process are organized in to seven major workflows:

- 1) Management
- 2) Environment
- 3) Requirements
- 4) Design
- 5) Implementation
- 6) Assessment
- 7) Deployment

Management workflow: controlling the process and ensuring win conditions for all stakeholders.

Environment workflow: automating the process and evolving the maintenance environment.

Requirements workflow: analyzing the problem space and evolving the requirements artifacts.

Design workflow: modeling the solution and evolving the architecture and design artifacts.

Implementation workflow: programming the components and evolving the implementation and deployment artifacts.

Assessment workflow: assessing the trends in process and product quality.

Deployment workflow: transitioning the end products to the user.

Table 2.4 The ARTIFACTS and life-cycle emphases associated with each workflow

WORKFLOW	ARTIFACTS	LIFE CYCLE PHASE EMPHASIS
Management	Business case Software development plan Status assessments Vision Work breakdown structure	Inception: Prepare business case and vision Elaboration: Plan development Construction: Monitor and control development Transition: Monitor and control deployment
Environment	Environment Software change order Database	Inception: Define development environment and change management infrastructure Elaboration: Install development environment and establish change management database Construction: Maintain development environment and software change order database Transition: Transition maintenance environment and software change order database

Requirements	Requirements set Release specifications Vision	Inception: Define operational concept Elaboration: Define architecture objectives Construction: Define iteration objectives Transition: Refine release objectives
Design	Design set Architecture description	Inception: Formulate architecture concept Elaboration: Achieve architecture baseline Construction: Design Components Transition: Refine architecture and components
Implementation	Implementation Set Deployment Set	Inception: Support architecture prototypes Elaboration: Produce architecture baseline Construction: Produce complete Component Transition: Maintain components
Assessment	Release specifications Release Descriptions User manuals Deployment set	Inception: Assess plans, vision, prototypes Elaboration: Assess architecture Construction: Assess internal releases Transition: Assess producer releases
Deployment	Deployment Set	Inception: Analyze user community Elaboration: Design user manual Construction: Prepare transition materials Transition: Transition product to user

Checkpoints of the Process:

It is important to place visible milestones in the life cycle in order to discuss the progress of the project by the Stakeholders and also to achieve,

- 1) Synchronize stakeholder expectations and achieve agreement among the requirements, the design, and the plan perspectives.
- 2) Synchronize related artifacts into a consistent and balanced state.
- 3) Identify the important risks, issues, and out-of-tolerance conditions.
- 4) Perform a global review for the whole life cycle, not just the current situation of an individual perspective or intermediate product.

Three sequences of project checkpoints are used to synchronize stakeholder expectations throughout the Lifecycle:

- 1) **Major milestones**
- 2) **Minor milestones**
- 3) **Status assessments**

1) Major Milestones:

The most important major milestone is usually the event that transitions the project from the elaboration phase into the construction phase. These are the system wide events are held at the end of each development phase. They provide visibility to system wide issues. Major milestones at the end of each phase use formal, stakeholder-approved evaluation criteria and release descriptions. In an iterative model, the major milestones are used to achieve concurrence among all stakeholders on the current state of the project. Different stakeholders have different concerns:

Customers: schedule and budget estimates, feasibility, risk assessment, requirements understanding, progress, product line compatibility.

Users: consistency with requirements and usage scenarios, potential for accommodating growth, quality attributes.

Architects and systems engineers: product line compatibility, requirements changes, tradeoff analyses, completeness and consistency, balance among risk, quality and usability.

Developers: Sufficiency of requirements detail and usage scenario descriptions, frameworks for component selection or development, resolution of development risk, product line compatibility, sufficiency of the development environment.

Maintainers: sufficiency of product and documentation artifacts, understandability, interoperability with existing systems, sufficiency of maintenance environment.

Others: regulatory agencies, independent verification and validation contractors, venture capital investors, subcontractors, associate contractors, and sales and marketing teams.

2) Minor Milestones: The format and content of minor milestones are highly dependent on the project and the organizational culture. These are the iteration-focused events are conducted to review the content of an iteration in detail and to authorize continued work. Minor milestones capture two artifacts: a release specification and a release description. Minor milestones use informal, development-team-controlled versions of these artifacts.

- The number of iteration-specific, informal milestones needed depends on the content and length of the iteration.
- Iterations which have one-month to six-month duration have only two milestones are needed: the iteration readiness review and iteration assessment review. For longer iterations some other intermediate review points are added.
- All iterations are not created equal . An iteration take different forms and priorities, depending on where the project is in the life cycle.
- Early iterations focus on analysis and design. Later iterations focus on completeness, consistency, usability and change management.
- **Iteration Readiness Review:** This informal milestone is conducted at the start of each iteration to review the detailed iteration plan and the evaluation criteria that have been allocated to this iteration.
- **Iteration Assessment Review:** This informal milestone is conducted at the end of each iteration to assess the degree to which the iteration achieved its objectives and to review iteration results, test results, to determine amount of rework to be done, to review impact of the iteration results on the plan for subsequent iterations.

3) Status Assessments: Periodic status assessments are important for focusing continuous attention on the evolving health of the project and its dynamic priorities. These are periodic events provide management with frequent and regular insight into the progress being made. These are management reviews conducted at regular intervals (monthly, quarterly) to address progress and quality of project and maintain open communication among all stakeholders. The main objective of this assessment is to synchronize all Stakeholders expectations and also serve as project snapshots. Also provide,

- 1) A mechanism for openly addressing, communicating, and resolving management issues, technical issues, and project risks.
- 2) A mechanism for broadcast process, progress, quality trends, practices, and experience information to and from all stakeholders in an open forum.
- 3) Objective data derived directly from on-going activities and evolving product configurations.



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