

Question Paper Solution (75:25), April 2015
Subject : Software Project Management

Ques1. (a) Discuss the significance, of reducing the product size, on ROI (returns on investment).
Explain, briefly, how the product size can be reduced.

Significance (1 mark)

- One important aspect of software economics is that the relationship between effort and size exhibits a diseconomy of scale.
- The more software you build, the more expensive it is per unit item.
- For e.g. 10, 000 line software solution will cost less per line than a 100,000 line software solution.
- Hence, the most significant way to improve ROI is usually to produce a product that achieves the goal with the minimum possible human-generated source material.

Ways to reduce the product size (4 marks)

1. Code Reuse
2. Object oriented technology
3. Automatic code generation / ready to use components
4. HLL

(brief explanation of above points)

Ques1(b) Discuss the key practices that improve overall software quality.

Key practices to improve overall software quality are:-

1. Focusing on requirements and critical use cases early in the life cycle.
2. Using metrics and indicators to measure the progress and quality.
3. Providing integrated life-cycle environment.
4. Using visual modeling and higher level languages.
5. Early and continuous insight into performance issues.

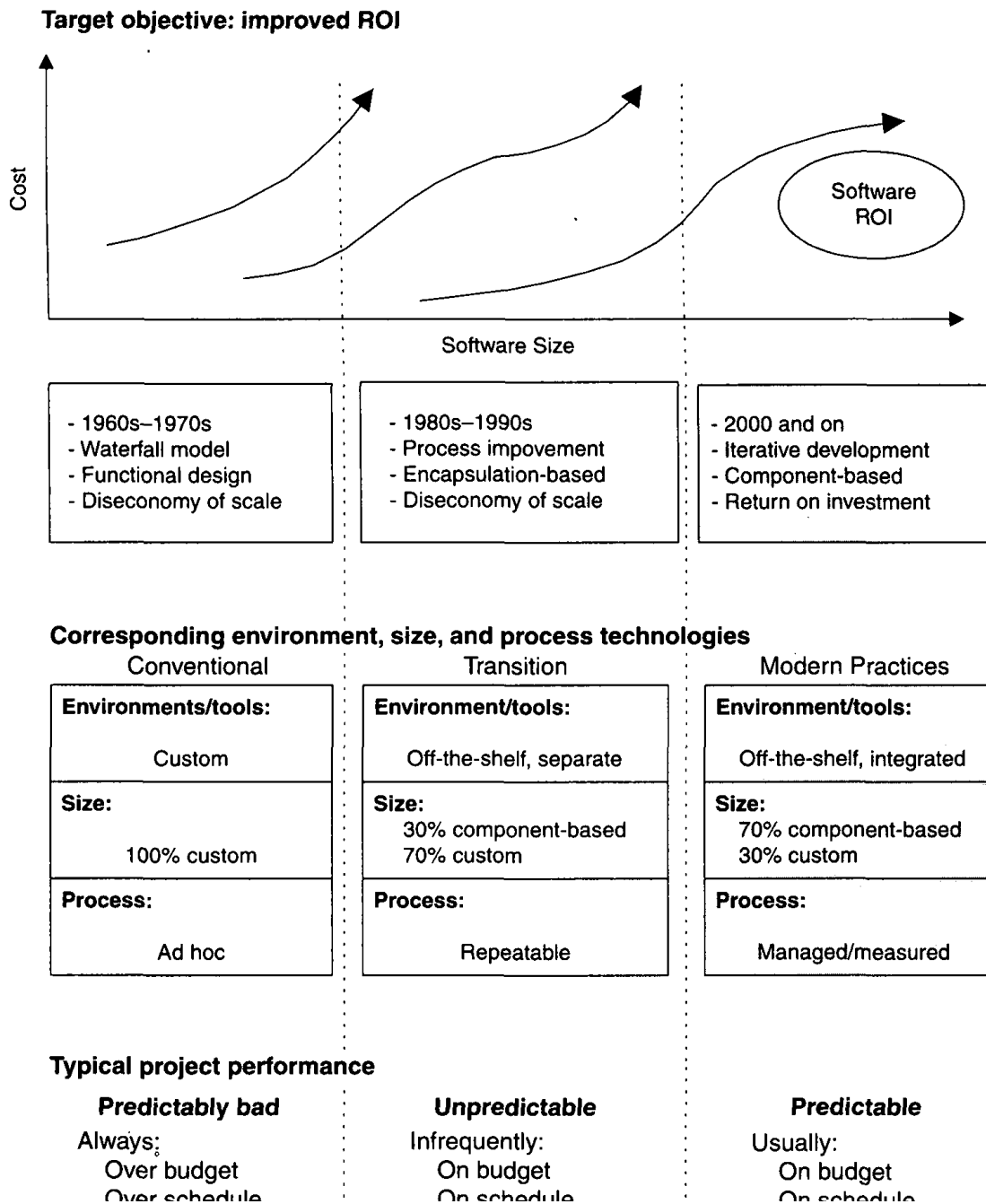
(brief explanation of all above points; 1 M for each point)

Ques1(c) Boehm's staffing principles :-

(brief explanation of each point carries 1 Mark)

1. Principle of Top Talent.
2. The principle of job matching
3. The principle of career progression
4. The principle of team balance
5. The principle of phase out

Ques1(d) Evolution of software economics over three generations.



Ques2 (a). Discuss the Life cycle, defined for modern software development process.

Two stages of the life-cycle :

1. The engineering stage – driven by smaller teams doing design and synthesis activities
2. The production stage – driven by larger teams doing construction, test, and deployment activities

Life-Cycle Phases :-

- Engineering Stage has following phases :-
 - o Inception Phase
 - o Elaboration Phase.
- Production Stage has following phases:-
 - o Construction Phase
 - o Transition Phase

Inception Phase - goal – to achieve concurrence among stakeholders on the life-cycle objectives

Elaboration Phase - During the elaboration phase, an executable architecture prototype is built

Construction Phase - All remaining components and application features are integrated into the application. All features are thoroughly tested

Transition Phase - The transition phase is entered when baseline is mature enough to be deployed in the end-user domain. This phase could include beta testing, conversion of operational databases, and training of users and maintainers.

Ques2(b) Define the term “Artifacts”. List the five sets of artifacts. Define the following: - Vision document, Software architecture description & release specifications.

Project Artifacts def. (1 Mark)

Project Artifacts are the lowest levels of project document-based objects (diagrams, design schemes, templates, agendas) that explore project work by phases and determine what results to produce upon completion of each phase. They define and document a planned outcome to be delivered under preset requirements and specifications.

Artifacts create project documentation. They are generated by the team throughout the project lifecycle. Each activity creates an artifact that documents a deliverable. All activity deliverables are defined by project artifacts.

Listing Five Sets (1 mark)

Requirements Set	Design Set	Implementation Set	Deployment Set
1. Vision document 2. Requirements model(s)	1. Design model(s) 2. Test model 3. Software architecture description	1. Source code baselines 2. Associated compile-time files 3. Component executables	1. Integrated product executable baselines 2. Associated run-time files 3. User manual

Planning Artifacts	Management Set	Operational Artifacts
1. Work breakdown structure 2. Business case 3. Release specifications 4. Software development plan		5. Release descriptions 6. Status assessments 7. Software change order database 8. Deployment documents 9. Environment

- **Vision document (1 Mark)** : vision statement (or user need) - which captures the contract between the development group and the buyer.
- **Software architecture description (1 mark)** : Architecture Description : it is extracted from the design model and includes views of the design, implementation, and deployment sets sufficient to understand how the operational concept of the requirements set will be achieved.
- **Release specifications (1 mark)**: It mainly contains Evaluation criteria. Evaluation criteria are the snapshots of objectives for a given intermediate life- cycle milestone.

Ques. 2c. Discuss three different aspects of software architecture from management perspective.

From a management perspective, there are three different aspects of an architecture : (3 marks)

1. An architecture (the intangible design concept) is the design of software system, as opposed to design of a component.
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 can be achieved within the parameters of the business case (cost, profit, time, people).
3. An architecture description (a human-readable representation of an architecture) is an organized subsets of information extracted from the design set model.

The importance of software architecture can be summarized as follows: **(2 marks)**

- Architecture representations provide a basis for balancing the trade-offs between the problem space and the solution space.
- Poor architectures and immature processes are often given as reasons for project failures.
- A mature process, an understanding of the primary requirements, and a demonstrable architecture are important prerequisites for predictable planning.
- Architecture development and process definition are the intellectual steps that map the problem to a solution without violating the constraints.

Ques 2d. Map the process exponent parameters of COCOMO to top 10 principles of modern process.

Exponent parameters of COCOMO model are :-

1. Application precedentedness
2. Process flexibility
3. Architecture risk resolution
4. Team cohesion
5. Software process maturity

Mapping of each COCOMO parameter to modern principles carries 1mark.

Ques3a. What do you mean by workflow? Discuss briefly its types.

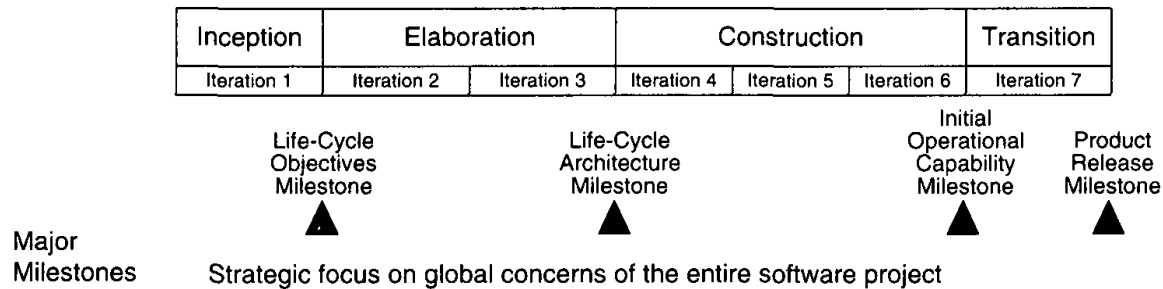
The term workflow is used to mean a thread of cohesive and most sequential activities.

There are seven top-level workflows:

1. Management workflow: controlling the process and ensuring win conditions for all stakeholders.
2. Environment workflow: automating the process and evolving the maintenance environment
3. Requirements workflow: analyzing the problem space and evolving the requirements artifacts
4. Design workflow: modeling the solution and evolving the architecture and design artifacts
5. Implementation workflow: programming the components and evolving the implementation and deployment artifacts
6. Assessment workflow: assessing the trends in process and product quality
7. Deployment workflow: transitioning the end products to the user

Ques 3b. Write a short note on Major Milestone.

Major milestones : provide visibility to systemwide issues, synchronize the management and engineering perspectives and verify that the aims of the phase have been achieved.



(brief explanation of four major milestones:- Life cycle objectives, Life cycle Architecture , Initial Operational capability & Product Release Milestone)

Ques 3c. Define Work breakdown structure. Give difference between conventional and evolutionary WBS. List issues related to conventional WBS.

DEF. : 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:

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

Difference between Conventional & evolutionary:- Main difference is that, An evolutionary WBS organizes the planning elements around the PROCESS Framework.

Conventional WBS is organized using PRODUCT framework.

Conventional work breakdown structures frequently suffer from three fundamental flaws.

1. They are prematurely structured around the product design.
2. They are prematurely decomposed, planned, and budgeted in either too much or too little detail.
3. They are project-specific, and cross-project comparisons are usually difficult or impossible.

Ques 3d.Explain forward looking approach for 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.

Ques 4a. Discuss briefly, default roles in a software Line-of-business organization.

Line of Business Organization: Line of Business Organizations The main features of the default organization are as follows. Responsibility for process definition and maintenance is specific to a cohesive line of business, where process commonality makes sense.

1. SEPA (Software Engineering Process Authority)
2. SEEA (Software Engineering Environment Authority)
3. PRA (Project Review Authority)
4. INFRASTRUCTURE

Line of Business Organization Def. – 1 mark

Each role brief explanation – 1 mark each

Ques 4b. Define Process Automation. Mention its significance. Also mention the extent of automation at each level of process.

Process Automation : Automation means the loss of many organization jobs. Automation needs growth depending on the scale of the effort. Process automation is critical to an iterative process.

There are many tools available to automate the software development process.

Significance :

Process automation is necessary to perform against the development plan with acceptable efficiency. Significant improvements of quality, shorter project durations and ultimately reduction of overall engineering costs can be achieved through Process automation. Hence increase in ROI.

The extent of automation at each level of process is as follows:-

1. **Metaprocess:** an organization's policies, procedures, and practices for managing a software-intensive line of business. The automation support for this level is called an *infrastructure*. An infrastructure is an inventory of preferred tools, artifact templates, microprocess guidelines, macroprocess guidelines, project performance repository, database of organizational skill sets, and library of precedent examples of past project plans and results.
2. **Macroprocess:** a project's policies, procedures, and practices for producing a complete software product within certain cost, schedule, and quality constraints. The automation support for a project's process is called an *environment*. An environment is a specific collection of tools to produce a specific set of artifacts as governed by a specific project plan.
3. **Microprocess:** a project team's policies, procedures, and practices for achieving an artifact of the software process. The automation support for generating an artifact is generally called a *tool*. Typical tools include requirements management, visual modeling, compilers, editors, debuggers, change management, metrics automation, document automation, test automation, cost estimation, and workflow automation.

Ques4c. Explain mapping between process workflows and software development tools.

There are many tools available to automate the software development process. The mapping of the software development tools to the process workflows is shown below:

Workflows	Environment tools and process automation			
Management	Workflow automation, metrics automation			
Environment	Change management, document automation			
Requirements	Requirements management			
Design	Visual modeling			
Implementation	Editor-compiler-debugger			
Assessment	Test automation, defect tracking			
Deployment	Defect tracking			
Process	Organization Policy			
Life Cycle	Inception	Elaboration	Construction	transition

Each of the process workflows has a distinct need for automation support.

Ques4d. “The project environment artifacts evolve through three discrete states”. Explain.

The project environment artifacts evolve through three discrete states: the prototyping environment, the development environment, and the maintenance environment.

1. *The prototyping environment* includes an architecture testbed for prototyping project architectures to evaluate trade-offs during the inception and elaboration phases of the life cycle. This informal configuration of tools should be capable of supporting the following activities:
 - Performance trade-offs and technical risk analyses
 - Make/buy trade-offs and feasibility studies for commercial products
 - Fault tolerance/dynamic reconfiguration trade-offs
 - Analysis of the risks associated with transitioning to full-scale implementation
 - Development of test scenarios, tools, and instrumentation suitable for analyzing the requirements
2. *The development environment* should include a full suite of development tools needed to support the various process workflows and to support round-trip engineering to the maximum extent possible.
3. *The maintenance environment* should typically coincide with a mature version of the development environment. In some cases, the maintenance environment may be a subset of the development environment delivered as one of the project's end products.

Ques5a. Define metrics and discuss characteristics of a good metric.

Metrics: Software process and project metrics are quantitative measures that enable software engineers to gain insight into the efficiency of the software process and the projects conducted using the process framework.

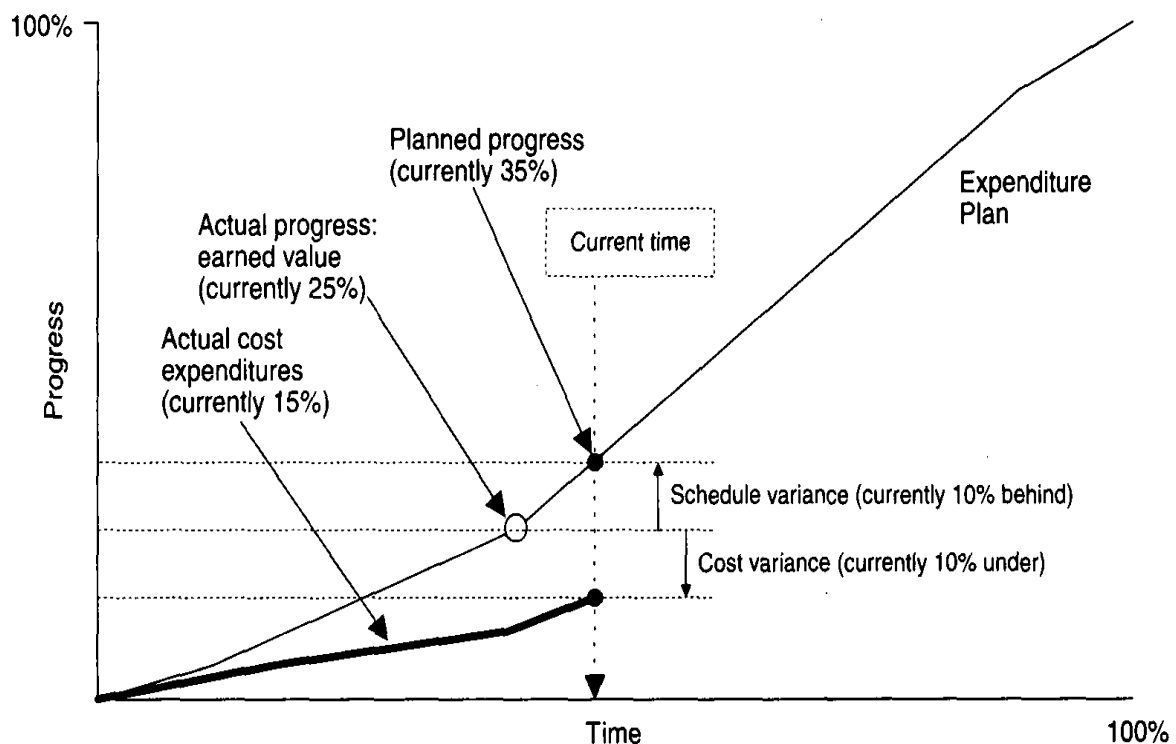
The basic characteristics of a good metric are as follows:

1. *It is considered meaningful by the customer, manager, and performer.* If any one of these stakeholders does not see the metric as meaningful, it will not be used. "The customer is always right" is a sales motto, not an engineering tenet. Customers come to software engineering providers because the providers are more expert than they are at developing and managing software. Customers will accept metrics that are demonstrated to be meaningful to the developer.
2. *It demonstrates quantifiable correlation between process perturbations and business performance.* The only real organizational goals and objectives are financial: cost reduction, revenue increase, and margin increase.
3. *It is objective and unambiguously defined.* Objectivity should translate into some form of numeric representation (such as numbers, percentages, ratios) as opposed to textual representations (such as excellent, good, fair, poor). Ambiguity is minimized through well-understood units of measurement (such as staff-month, SLOC, change, function point, class, scenario, requirement), which are surprisingly hard to define precisely in the software engineering world.
4. *It displays trends.* This is an important characteristic. Understanding the change in a metric's value with respect to time, subsequent projects, subsequent releases, and so forth is an extremely important perspective, especially for today's iterative development models. It is very rare that a given metric drives the appropriate action directly. More typically, a metric presents a perspective. It is up to the decision authority (manager, team, or other information processing entity) to interpret the metric and decide what action is necessary.
5. *It is a natural by-product of the process.* The metric does not introduce new artifacts or overhead activities; it is derived directly from the mainstream engineering and management workflows.
6. *It is supported by automation.* Experience has demonstrated that the most successful metrics are those that are collected and reported by automated tools, in part because software tools require rigorous definitions of the data they process.

Ques5b. Write a short note on “Earned Value System”.

Modern software processes are amenable to financial performance measurement through an earned value approach. The basic parameters of an earned value system usually expressed in units of dollars, are as follows:

- Expenditure plan: the planned spending profile for a project over its planned schedule. For most software projects (and other labor-intensive projects), this profile generally tracks the staffing profile.
- Actual progress: the technical accomplishment relative to the planned progress underlying the spending profile. In a healthy project, the actual progress tracks planned progress closely.
- Actual cost: the actual spending profile for a project over its actual schedule. In a healthy project, this profile tracks the planned profile closely.
- Earned value: the value that represents the planned cost of the actual progress.
- Cost variance: the difference between the actual cost and the earned value. Positive values correspond to over-budget situations; negative values correspond to under-budget situations.
- Schedule variance: the difference between the planned cost and the earned value. Positive values correspond to behind-schedule situations; negative values correspond to ahead-of-schedule situations.



Ques5c. Discuss the “Tailoring” concept in context of software development. Explain the two primary dimensions of process variability.

Tailoring concept (2 marks)

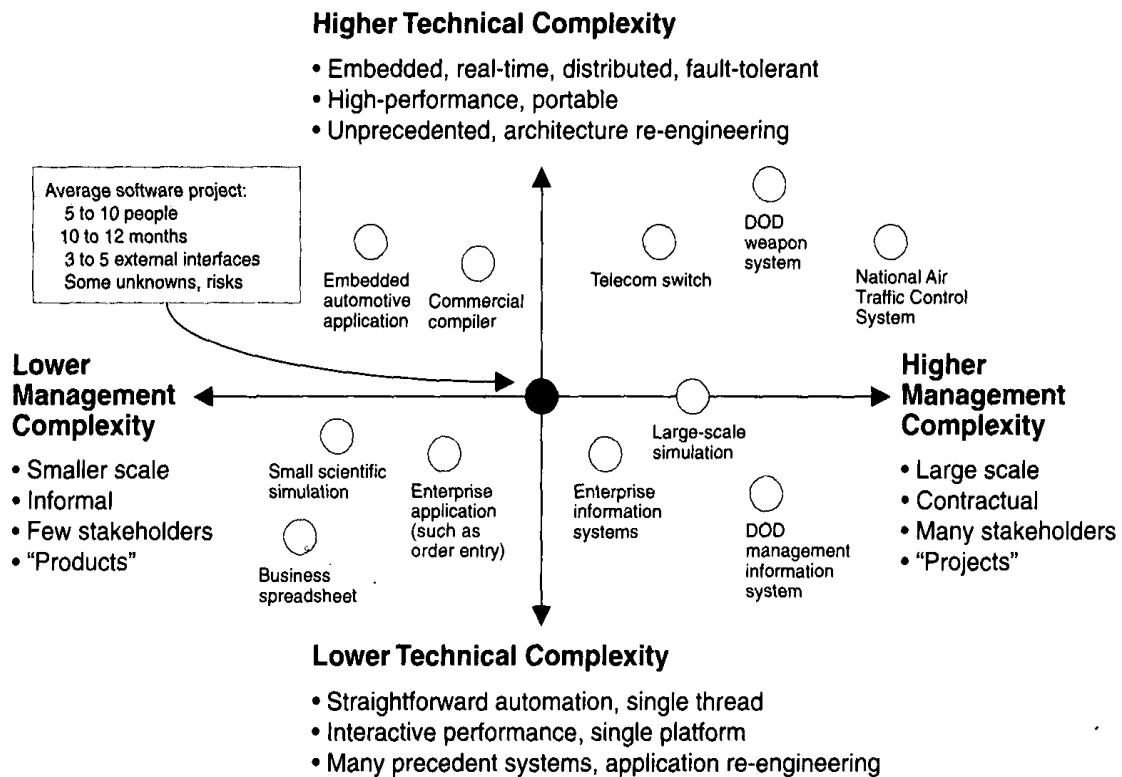
It is necessary to tailor the software management effort to the specific needs of project.

Tailoring in software development is the process of extracting a set of processes, task and artifacts from the organization established process, task and artifacts to achieve its objectives.

Two primary dimensions of process variability are :-

Technical Complexity

Management Complexity



(Either the above diagram or its explanation – 3 marks)

Ques5d. Explain the process discriminators resulting from differences in process maturity.

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, con- tent, and production methods	Free-form
Workflow effort allocations	Well-established basis	No basis
Checkpoints	Well-defined combination of formal and informal events	(insignificant)
Management discipline	Predictable planning Objective status assessments	Informal planning and project control
Automation discipline	Requires high levels of automa- tion for round-trip engineering, change management, and pro- cess instrumentation	Little automation or disconnected islands of automation

Ques6a. Discuss the five recurring issues of conventional process. How are they resolved by modern process framework?

1. *Protracted integration and late design breakage* are resolved by forcing integration into the engineering stage. This is achieved through continuous integration of an architecture baseline supported by executable demonstrations of the primary scenarios.
2. *Late risk resolution* is resolved by emphasizing an architecture-first approach, in

which the high-leverage elements of the system are elaborated early in the life cycle.

3. The analysis paralysis of a *requirements-driven functional decomposition* is avoided by organizing lower level specifications along the content of releases rather than along the product decomposition (by subsystem, by component, etc.).
4. *Adversarial stakeholder relationships* are avoided by providing much more tangible and objective results throughout the life cycle.
5. The conventional *focus on documents and review meetings* is replaced by a focus on demonstrable results and well-defined sets of artifacts, with more-rigorous notations and extensive automation supporting a paperless environment.

Ques6b. How is risk resolution carried out in the iterative process. What is its advantage?

According to modern project management ways, risk resolution is done very early in the life cycle.

The engineering stage of the life cycle focuses on exploring the risk and resolving them before production stage.

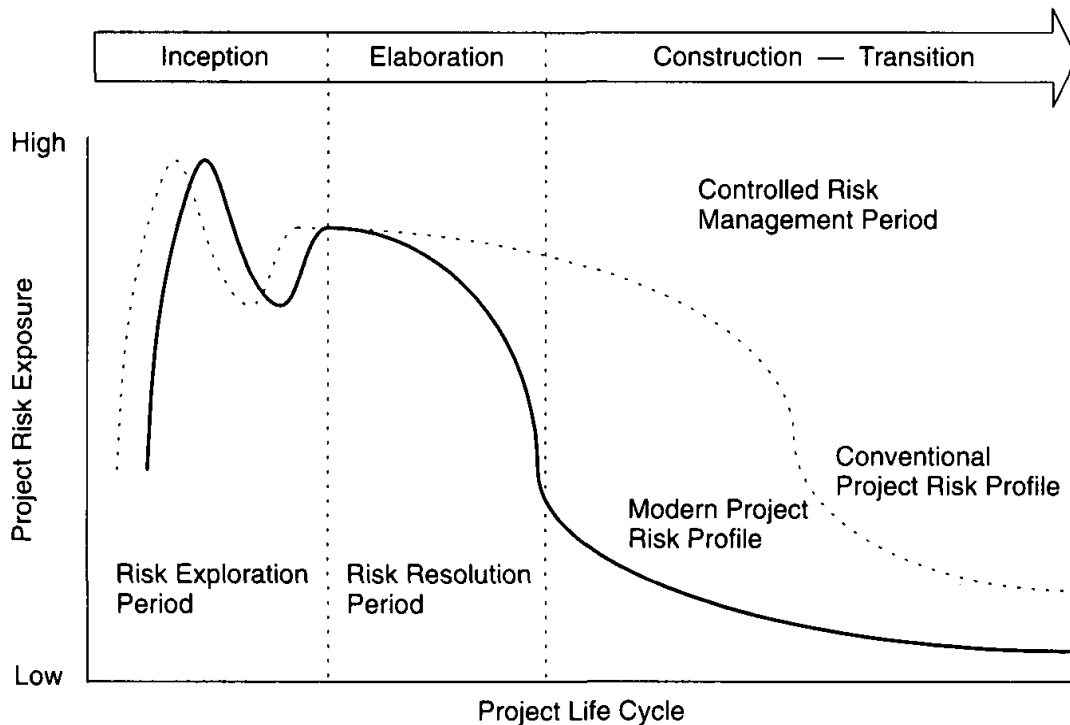
Entire project life cycle is divided into three different time period related to Risk resolution.

They are as follows:-

Risk Exploration Period (REP)

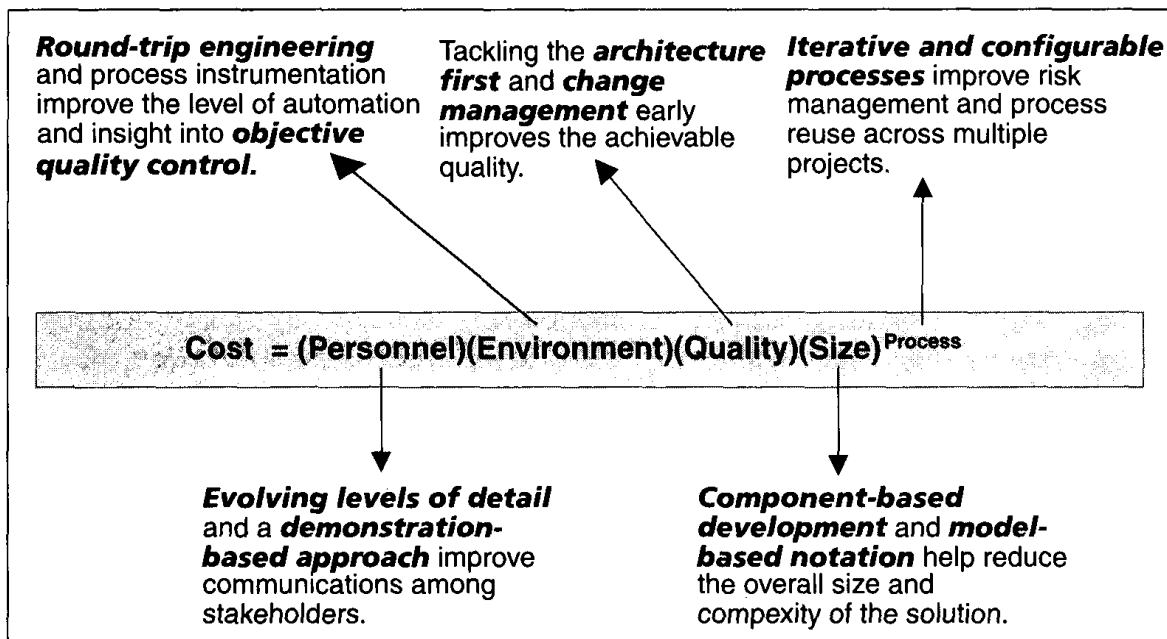
Risk Resolution Period (RRP)

Controlled Risk Management Period (CRMP)



Ques6c. How, balancing the top 10 software management principles, achieve balance in software economics equation.

(Following diagram with its explanation)



Ques6d. Discuss nine best practices of software management.

1. *Formal risk management.*

▲ Using an **iterative process** that confronts risk is more or less what this is saying.

2. *Agreement on interfaces.*

▲ While we may use different words, this is exactly the same intent as my **architecture-first** principle. Getting the architecture baselined forces the project to gain agreement on the various external interfaces and the important internal interfaces, all of which are inherent in the architecture.

3. *Formal inspections.*

▲ The assessment workflow throughout the life cycle, along with the other engineering workflows, must balance several different defect removal strategies. The least important strategy, in terms of breadth, should be formal inspection, because of its high costs in human resources and its low defect discovery rate for the critical architectural defects that span multiple components and temporal complexity.

4. *Metric-based scheduling and management.*

▲ This important principle is directly related to my **model-based notation** and **objective quality control** principles. Without rigorous notations for artifacts, the measurement of progress and quality degenerates into subjective estimates.

5. *Binary quality gates at the inch-pebble level.*

▲ This practice is easy to misinterpret. Too many projects have taken exactly this approach early in the life cycle and have laid out a highly detailed plan at great expense. Three months later, when some of the requirements change or the architecture changes, a large percentage of the detailed planning must be rebaselined. A better approach would be to maintain fidelity of the plan commensurate with an understanding of the requirements and the architecture. Rather than inch pebbles, I recommend establishing milestones in the engineering stage followed by inch pebbles in the production stage. This is the primary message behind my **evolving levels of detail** principle.

6. *Programwide visibility of progress versus plan.*

▲ This practice—namely, open communications among project team members—is obviously necessary. None of my principles traces directly to this practice. It seems so obvious, I let it go without saying.

7. *Defect tracking against quality targets.*

▲ This important principle is directly related to my **architecture-first** and **objective quality control** principles. The make-or-break defects and quality targets are architectural. Getting a handle on these qualities early and tracking their trends are requirements for success.

8. *Configuration management.*

▲ The Airlie Software Council emphasized configuration management as key to controlling complexity and tracking changes to all artifacts. It also recognized that automation is important because of the volume and dynamics of modern, large-scale projects, which make manual methods cost-prohibitive and error-prone. The same reasoning is behind my **change management** principle.

9. *People-aware management accountability.*

▲ This is another management principle that seems so obvious, I let it go without saying.

Ques7a. How “Peer Inspection” helps in improving ROI? Explain.

Peer Inspection : In software development, [peer review](#) is a type of [software review](#) in which a work product (document, code, or other) is examined by its author and one or more colleagues, in order to evaluate its technical content and quality.

Peer Inspection finds the problems very early in the project life cycle, hence improves the ROI. It provide a significant return. One value of inspections is in the professional development of a team. It is generally useful to have the products of junior team members reviewed by senior mentors. Putting the products of amateurs into the hands of experts and vice versa is a good mechanism for accelerating the acquisition of knowledge and skill in new personnel. Gross blunders can be caught and feedback can be appropriately channeled, so that bad practices are not perpetuated. This is one of the best ways for junior software engineers to learn.

Ques7b. Explain in detail “Transition phase” of software development life cycle with the following details: - primary objectives, essential activities & evaluation criteria.

The transition phase is entered when a baseline is mature enough to be deployed in the end-user domain. This typically requires that a usable subset of the system has been achieved with acceptable quality levels and user documentation so that transition to the user will provide positive results. This phase could include any of 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

The transition phase concludes when the deployment baseline has achieved the complete vision. For some projects, this life-cycle end point may coincide with the life-cycle starting point for the next version of the product. For others, it may coincide with a complete delivery of the information sets to a third party responsible for operation, maintenance, and enhancement.

The transition phase focuses on the activities required to place the software into the hands of the users. Typically, this phase includes several iterations, including beta releases, general availability releases, and bug-fix and enhancement releases. Considerable effort is expended in developing user-oriented documentation, training users, supporting users in their initial product use, and reacting to user feedback. (At this point in the life cycle, user feedback should be confined mostly to product tuning, configuring, installing, and usability issues.)

PRIMARY OBJECTIVES

- Achieving user self-supportability
- Achieving stakeholder concurrence that deployment baselines are complete and consistent with the evaluation criteria of the vision
- Achieving final product baselines as rapidly and cost-effectively as practical

ESSENTIAL ACTIVITIES

- Synchronization and integration of concurrent construction increments into consistent deployment baselines

- Deployment-specific engineering (cutover, commercial packaging and production, sales rollout kit development, field personnel training)
- Assessment of deployment baselines against the complete vision and acceptance criteria in the requirements set

EVALUATION CRITERIA

- Is the user satisfied?
- Are actual resource expenditures versus planned expenditures acceptable?

Ques7c. What is the significance of periodic assessments? Discuss the contents of Status assessment review.

Periodic status assessments serve as project snapshots. While the period may vary, the recurring event forces the project history to be captured and documented. Status assessments provide the following:

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

TOPIC	CONTENT
Personnel	Staffing plan vs. actuals Attritions, additions
Financial trends	Expenditure plan vs. actuals for the previous, current, and next major milestones Revenue forecasts
Top 10 risks	Issues and criticality resolution plans Quantification (cost, time, quality) of exposure
Technical progress	Configuration baseline schedules for major milestones Software management metrics and indicators Current change trends Test and quality assessments
Major milestone plans and results	Plan, schedule, and risks for the next major milestone Pass/fail results for all acceptance criteria
Total product scope	Total size, growth, and acceptance criteria perturbations

Ques7d. Discuss the primitive components of a software change order.

The basic fields of the SCO are title, description, metrics, resolution, assessment and disposition.

- *Title.* The title is suggested by the originator and is finalized upon acceptance by the configuration control board (CCB). This field should include a reference to an external software problem report if the change was initiated by an external person (such as a user).
- *Description.* The problem description includes the name of the originator, date of origination, CCB-assigned SCO identifier, and relevant version identifiers of related support software. The textual problem description should provide as much detail as possible, along with attached code excerpts, display snapshots, error messages, and any other data that may help to isolate the problem or describe the change needed.
- *Metrics.* The metrics collected for each SCO are important for planning, for scheduling, and for assessing quality improvement. Change categories are type 0 (critical bug), type 1 (bug), type 2 (enhancement), type 3 (new feature), and type 4 (other), as described later in this section. Upon acceptance of the SCO, initial estimates are made of the amount of breakage and the effort required to resolve the problem. The *breakage* item quantifies the
- *Resolution.* This field includes the name of the person responsible for implementing the change, the components changed, the actual metrics, and a description of the change. Although the level of component fidelity with which a project tracks change references can be tailored, in general, the lowest level of component references should be kept at approximately the level of allocation to an individual. For example, a “component” that is allocated to a team is not a sufficiently detailed reference.
- *Assessment.* This field describes the assessment technique as either inspection, analysis, demonstration, or test. Where applicable, it should also reference all existing test cases and new test cases executed, and it should identify all different test configurations, such as platforms, topologies, and compilers.
- *Disposition.* The SCO is assigned one of the following states by the CCB:
 - Proposed: written, pending CCB review
 - Accepted: CCB-approved for resolution
 - Rejected: closed, with rationale, such as not a problem, duplicate, obsolete change, resolved by another SCO
 - Archived: accepted but postponed until a later release
 - In progress: assigned and actively being resolved by the development organization
- In assessment: resolved by the development organization; being assessed by a test organization
- Closed: completely resolved, with the concurrence of all CCB members

Ques7e. Write a short note on SPCP(software project control panel).

The SPCP is one example of a metrics automation approach that collects, organizes, and reports values and trends extracted directly from the evolving engineering artifacts. Software engineers will accept metrics only if metrics are automated by the environment.

- Start the SPCP. The SPCP starts and shows the most current information that was saved when the user last used the SPCP.
- Select a panel preference. The user selects from a list of previously defined default panel preferences. The SPCP displays the preference selected.
- Select a value or graph metric. The user selects whether the metric should be displayed for a given point in time or in a graph, as a trend. The default for values is the most recent measurement available. The default for trends is monthly.
- Select to superimpose controls. The user points to a graphical object and requests that the control values for that metric and point in time be displayed. In the case of trends, the controls are shown superimposed with the metric.
- Drill down to trend. The user points to a graphical object displaying a point in time and drills down to view the trend for the metric.
- Drill down to point in time. The user points to a graphical object displaying a trend and drills down to view the values for the metric.
- Drill down to lower levels of information. The user points to a graphical object displaying a point in time and drills down to view the next level of information.
- Drill down to lower level of indicators. The user points to a graphical object displaying an indicator and drills down to view the breakdown of the next level of indicators.

Ques7f. Write a short note on “Next Generation Cost Models”.

$$\text{Effort} = F(T_{\text{Arch}}, S_{\text{Arch}}, Q_{\text{Arch}}, P_{\text{Arch}}) + F(T_{\text{App}}, S_{\text{App}}, Q_{\text{App}}, P_{\text{App}})$$

$$\text{Time} = F(P_{\text{Arch}}, \text{Effort}_{\text{Arch}}) + F(P_{\text{App}}, \text{Effort}_{\text{App}})$$

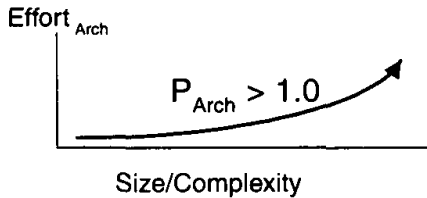
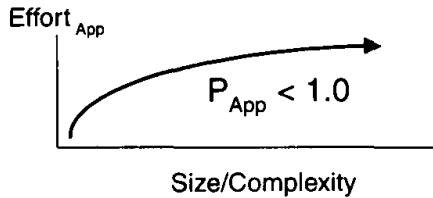
where:

T = technology parameter (environment automation support)

S = scale parameter (such as use cases, function points, source lines of code)

Q = quality parameter (such as portability, reliability, performance)

P = process parameter (such as maturity, domain experience)

Engineering Stage	Production Stage
<p>Risk resolution, low-fidelity plan Schedule/technology-driven Risk sharing contracts/funding</p>	<p>Low-risk, high-fidelity plan Cost-driven Fixed-price contracts/funding</p>
<p>N-month design phase</p>	<p>M-month production increments</p>
	
<p>Team Size Architecture: small team of software engineers Applications: small team of domain engineers Small and expert as possible</p>	<p>Team Size Architecture: small team of software engineers Applications: as many as needed Large and diverse as needed</p>
<p>Product Executable architecture Production plans Requirements</p>	<p>Product Deliverable, useful function Tested baselines Warranted quality</p>
<p>Focus Design and integration Host development environment</p>	<p>Focus Implement, test, and maintain Target technology</p>
<p>Phases Inception and elaboration</p>	<p>Phases Construction and transition</p>