(2½ hours)

Total Marks: 75

N. B.: (1) **All** questions are **compulsory**.
(2) Make **suitable assumptions** wherever necessary and **state the assumptions** made.
(3) Answers to the **same question** must be **written together**.
(4) Numbers to the **right** indicate **marks**.
(5) Draw **neat labeled diagrams** wherever **necessary**.
(6) Use of **Non-programmable** calculators is **allowed**.

1. Attempt *any two* of the following: **10**

a. What are the factors that can be abstracted with software economics? Explain in detail.

SPM-Walker Royce(pg no 21 &22)
Enlist the five factors: (1 mark)
Most software cost models can be abstracted into a function of five basic parameters:

- **size,**
- **process,**
- **personnel,**
- **environment, and**
- **quality.**

Explanation of all the five in brief: (3 marks)

The size of the end product (in human□generated components), which is typically quantified in terms of the number of source instructions or the number of function points required to develop the required functionality

The process used to produce the end product, in particular the ability of the process to avoid non-value-adding activities (rework, bureaucratic delays, communications overhead)

The capabilities of software engineering personnel, and particularly their experience with the computer science issues and the applications domain issues of the project.

The environment, which is made up of the tools and techniques available to support efficient software development and to automate the process

The required quality of the product, including its features, performance, reliability, and adaptability

Formula: (1 mark)

The relationships among these parameters and the estimated cost can be written as follows:
Effort = (Personnel)(Environment)(Quality)(Size Process)

b. What is customization of component & compare them with commercial components.
SPM—Walker Royce (pg. 39 & 40)
10 points--- 5 marks

TABLE 3-3. *Advantages and disadvantages of commercial components versus custom software*

APPROACH	ADVANTAGES	DISADVANTAGES
Commercial components	Predictable license costs Broadly used, mature technology Available now Dedicated support organization Hardware/software independence Rich in functionality	Frequent upgrades Up-front license fees Recurring maintenance fees Dependency on vendor Run-time efficiency sacrifices Functionality constraints Integration not always trivial No control over upgrades and maintenance Unnecessary features that consume extra resources Often inadequate reliability and stability Multiple-vendor incompatibilities
Custom development	Complete change freedom Smaller, often simpler implementations Often better performance Control of development and enhancement	Expensive, unpredictable development Unpredictable availability date Undefined maintenance model Often immature and fragile Single-platform dependency Drain on expert resources

c.

SPM---Walker Royce(Pg. 40 & 41)
Three perspectives--- Meta, Micro and Macro processes. Three paragraphs or tables that cover 5 points such as (5 points---5 marks)
Objectives, Audience, Metrics, concerns and timescale

TABLE 3-4. *Three levels of process and their attributes*

ATTRIBUTES	METAPROCESS	MACROPROCESS	MICROPROCESS
Subject	Line of business	Project	Iteration
Objectives	Line-of-business profitability Competitiveness	Project profitability Risk management Project budget, schedule, quality	Resource management Risk resolution Milestone budget, schedule, quality
Audience	Acquisition authorities, customers Organizational management	Software project managers Software engineers	Subproject managers Software engineers
Metrics	Project predictability Revenue, market share	On budget, on schedule Major milestone success Project scrap and rework	On budget, on schedule Major milestone progress Release/iteration scrap and rework
Concerns	Bureaucracy vs. standardization	Quality vs. financial performance	Content vs. schedule
Time scales	6 to 12 months	1 to many years	1 to 6 months

d.

Compare the three generations of software development.

SPM---Walker Royce (Pg No 23 & 24)

3 Generations:

Conventional

Transition

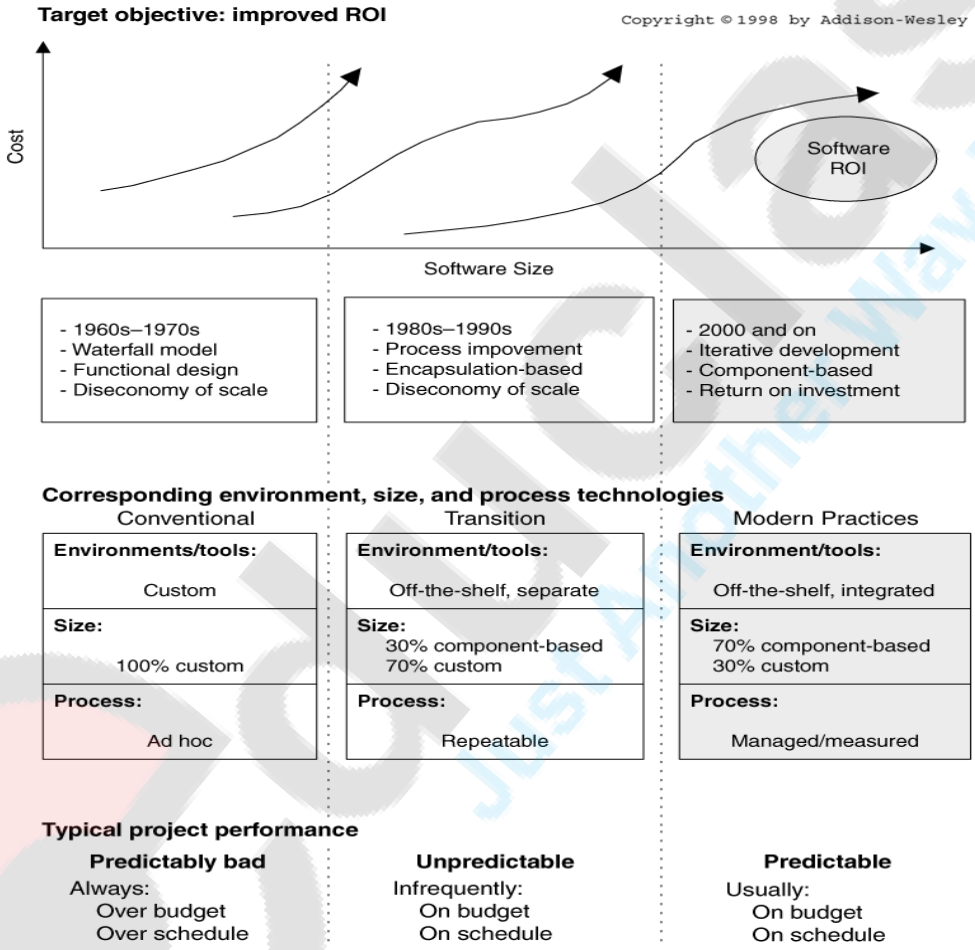
Modern

10 parameters such as

1. cost
2. size
3. ROI
4. Tools
5. Process
6. Methodology
7. Concept

8. Period
9. Objective
10. Project Performance

to be explained. (10 points---5 marks)



2. Attempt *any two* of the following: **10**

a. "Management set artifacts associated with process planning & execution"- justify the statement.

SPM---Walker Royce (Pg No: 89 & 86)

What is Management set of Artifact? (1 mark)

Its main focus: (1 mark)

Set of Artifacts: (Enlist and brief explanation --2 marks)

Diagram that shows the artifacts (1 mark)

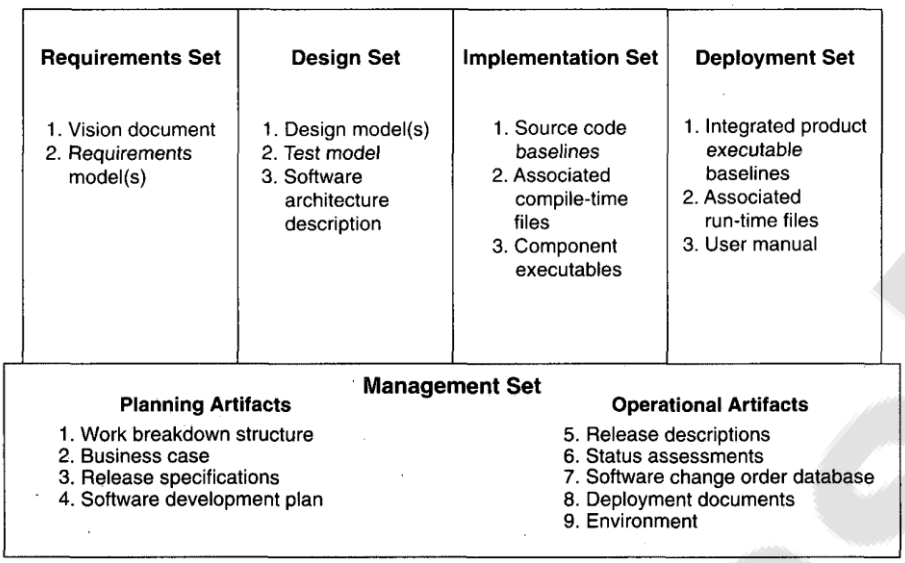


FIGURE 6-1. *Overview of the artifact sets*

b.

Write a short note on test artifacts.

SPM-- walker Royce (Pg No: 94)

What is test artifact (1 mark)
Need of test artifacts(1 mark)
Five points that helps to convert artifacts into processes. (3 marks)

product development. In essence, we are simply identifying the test infrastructure necessary to execute the test process as a required subset of the end product. By doing this, we have forced several engineering disciplines into the process.

- The test artifacts must be developed concurrently with the product from inception through deployment. Thus, testing is a full-life-cycle activity, not a late life-cycle activity.
- The test artifacts are communicated, engineered, and developed within the same artifact sets as the developed product.
- The test artifacts are implemented in programmable and repeatable formats (as software programs).
- The test artifacts are documented in the same way that the product is documented.
- Developers of the test artifacts use the same tools, techniques, and training as the software engineers developing the product.

These disciplines allow for significant levels of homogenization across project workflows, which are described in Chapter 8. Everyone works within the notations and techniques of the four sets used for engineering artifacts, rather than with separate sequences of design and test documents. Interpersonal communications, stakeholder reviews, and engineering analyses can be performed with fewer distinct formats, fewer ad hoc notations, less ambiguity, and higher efficiency.

c. Explain the technical perspective of software architecture.

SPM---Walker Royce(Pg No: 111-113)
Description:---(1 mark)
Models: ---(1 mark)
Four views: ---(2 marks)
Diagram: ---(1 mark)

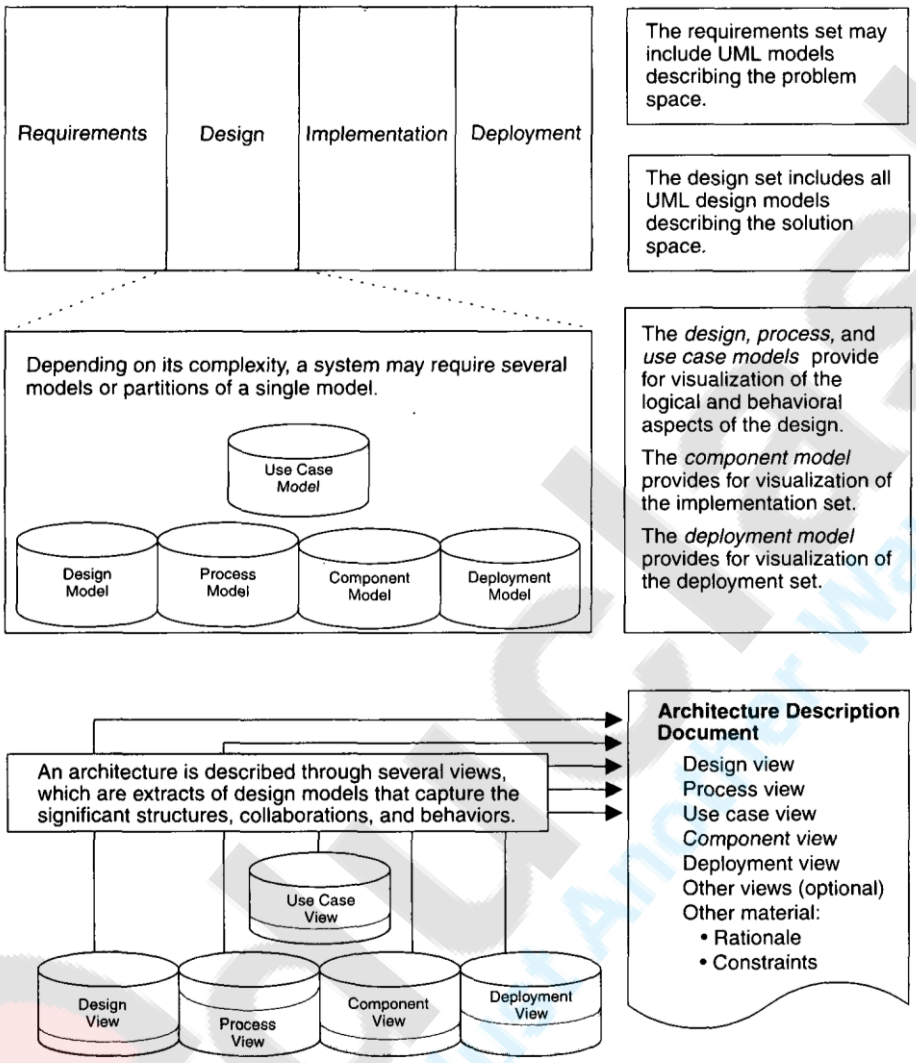


FIGURE 7-1. *Architecture, an organized and abstracted view into the design models*

d. Why do we call the requirements set the primary engineering content? Explain.

SPM-- Walker Royce (Pg no 86)

What is Requirement set of Artifact? (1 mark)
Its main focus: (1 mark)
Set of Artifacts: (Enlist and brief explanation --2 marks)
Diagram that shows artifacts: (1 mark)

6.1.2 The Engineering Sets

The engineering sets consist of the requirements set, the design set, the implementation set, and the deployment set. The primary mechanism for evaluating the evolving quality of each artifact set is the transitioning of information from set to set, thereby maintaining a balance of understanding among the requirements, design, implementation, and deployment artifacts. Each of these components of the system description evolves over time.

Requirements Set

Structured text is used for the vision statement, which documents the project scope that supports the contract between the funding authority and the project team. Ad hoc formats may also be used for supplementary specifications (such as regulatory requirements) and user mockups or other prototypes that capture requirements. UML notation is used for engineering representations of requirements models (use case models, domain models). The requirements set is the primary engineering context for evaluating the other three engineering artifact sets and is the basis for test cases.

Requirements artifacts are evaluated, assessed, and measured through a combination of the following:

- Analysis of consistency with the release specifications of the management set
- Analysis of consistency between the vision and the requirements models
- Mapping against the design, implementation, and deployment sets to evaluate the consistency and completeness and the semantic balance between information in the different sets
- Analysis of changes between the current version of requirements artifacts and previous versions (scrap, rework, and defect elimination trends)
- Subjective review of other dimensions of quality

---

3. Attempt *any two* of the following: **10**

a. Write a short note on realistic planning.

SPM---Walker Royce(Page no: 153)
Points to be explained: (10 points---5 marks)

1. Planning, requirements and architecture are the three perspectives.
2. The end products with these perspectives are a software development plan, a requirements specification, and an architecture description document.
3. On most successful projects, these three products are not very important once they have been produced.
4. HOWEVER, The act of planning is extremely important to project success.
5. It provides a framework and forcing function for making decisions.
6. It ensures buy-in by stakeholders and performers.
7. It transforms subjective, generic process frameworks into objective processes.
8. Inadequate planning are the most important reasons for failure of a project.
9. Finally, plans are not just for managers.
10. An open and visible planning process results in more ownership among all team members who execute the plan.

b. How do you classify the joint management reviews ? Explain about minor milestones in detail.

SPM---Walker Royce (Pg: 126 & 132)
What is joint-management reviews? (1 mark)
Classification: (1 mark)

- **Major milestones**
- **Minor milestones**

- **Status assessments**

About Minor Mile stones---in brief(2 marks)

Brief explanation of Two reviews: ---Iteration readiness and iteration assessment review

Typical minor milestone diagram: (1 mark)

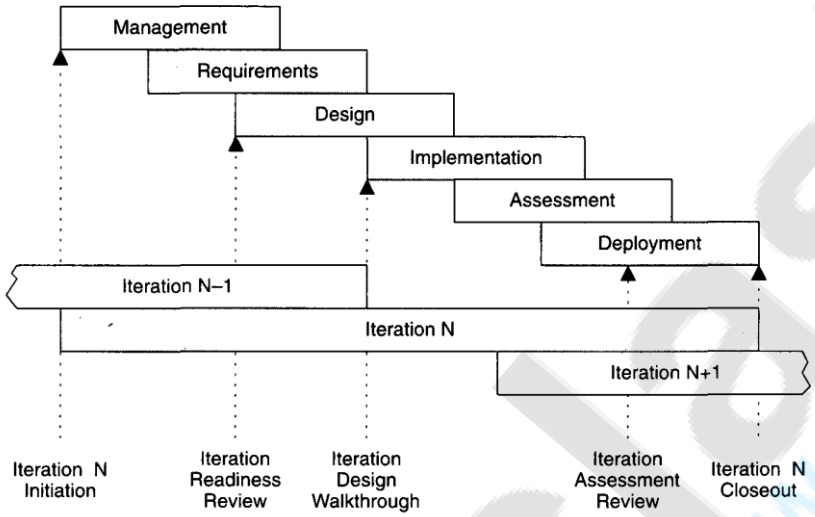


FIGURE 9-4. *Typical minor milestones in the life cycle of an iteration*

c. Explain the concept of interaction work flow in detail.

SPM ---Walker Royce (121-123)

Iteration workflow can be termed as Interaction workflow as they ensure cohesiveness between activities ; it can also be termed as Integration workflow as the developed processes should be integrated and it can also be termed as Incremental Workflow as it produces short releases as its output.
What is iteration workflow---1 mark
Workflow of an iteration--- diagram and explanation---2 marks
Iteraction across life-cycle phases---2 marks

8.2 ITERATION WORKFLOWS

An iteration consists of a loosely sequential set of activities in various proportions, depending on where the iteration is located in the development cycle. Each iteration is defined in terms of a set of allocated usage scenarios. The components needed to implement all selected scenarios are developed and integrated with the results of previous iterations. An individual iteration's workflow, illustrated in Figure 8-2, generally includes the following sequence:

- Management: iteration planning to determine the content of the release and develop the detailed plan for the iteration; assignment of work packages, or tasks, to the development team
- Environment: evolving the software change order database to reflect all new baselines and changes to existing baselines for all product, test, and environment components

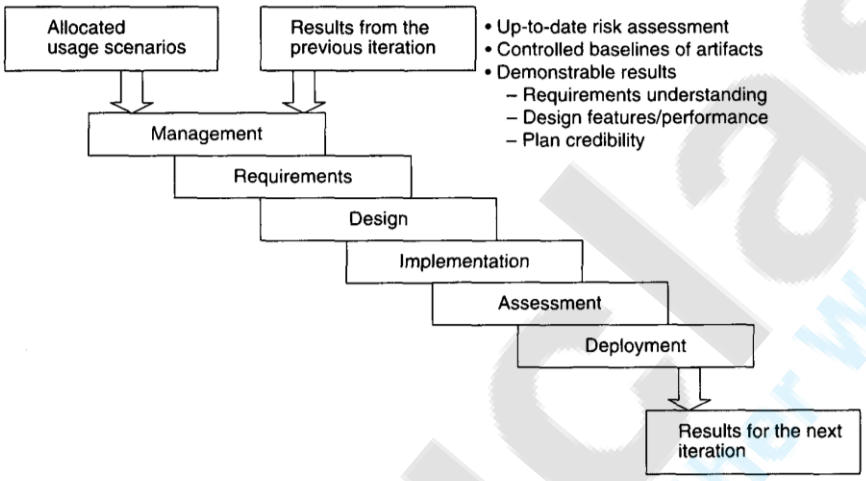


FIGURE 8-2. *The workflow of an iteration*

d. How to create a tailor-made WBS? Discuss the ways and means.

SPM---Walker Royce---(Pg. 142 & 143)
WBS and their classifications: (1 mark) (Text or diagrammatic representation)
What is Evolutionary WBS? (1 mark)
Five factors that help in constructing: (3 mark)

1. **Scale**
2. **Organizational Structure**
3. **Degree of Custom Development**
4. **Business Context and**
5. **Precedent Experience**

Brief explanation:

Scale

Larger projects have more levels.

Organizational Structure

Subcontractors or associate contractors require additional management subelements.

Degree of Custom Development

Fully custom requires more in design and implementation to manage risks.

Business Context

Large contractual projects generally require additional levels for management and accounting purposes.

Precedent Experience

Projects should exploit existing work and WBS development.

4. Attempt *any two* of the following: **10**

a. Explain about Project Software Standard.

SPM---Walker Royce---(Page no: 181)

Tools and organization policy helps in setting standard for software development.

Introduction about organization policy : (1 mark)

General Questions to be asked before setting organization policy: (1 mark)

Recurring Themes of Organizational Policy: (1 mark)

Various levels of Organizational policy: (2 marks)

- **Highest**
- ***I*ntermediate and**
- **Lowest Levels**

12.2.3 INFRASTRUCTURES

From a process automation perspective, the organization's infrastructure provides the organization's capital assets, including two key artifacts: a policy that captures the standards for project software development processes, and an environment that captures an inventory of tools. These tools are the automation building blocks from which project environments can be configured efficiently and economically.

Organization Policy

The organization policy is usually packaged as a handbook that defines the life cycle and the process primitives (major milestones, intermediate artifacts, engineering repositories, metrics, roles and responsibilities). The handbook provides a general framework for answering the following questions:

- What gets done? (activities and artifacts)
- When does it get done? (mapping to the life-cycle phases and milestones)
- Who does it? (team roles and responsibilities)
- How do we know that it is adequate? (checkpoints, metrics, and standards of performance)

The last point deserves further discussion. There are many different organizational structures throughout the software development industry. Most software-intensive companies have three distinct levels of organization, with a different policy focus at each level:

- Highest organization level: standards that promote (1) strategic and long-term process improvements, (2) general technology insertion and education, (3) comparability of project and business unit performance, and (4) mandatory quality control
- Intermediate line-of-business level: standards that promote (1) tactical and short-term process improvement, (2) domain-specific technology insertion and training, (3) reuse of components, processes, training, tools, and personnel experience, and (4) compliance with organizational standards
- Lowest project level: standards that promote (1) efficiency in achieving quality, schedule, and cost targets, (2) project-specific training, (3) compliance with customer requirements, and (4) compliance with organizational/business unit standards

b. What are the key points that software architecture team should focus on? Explain their roles, artifacts and responsibilities.

SPM---Walker Royce (Page No: 159 & 161)

Focusing points of software architecture team: with diagram(2 marks)

Roles (1 mark)

Artifacts (1 mark)

Responsibilities (1 mark)

Software Architecture Team

The software architecture team is responsible for the architecture. This responsibility encompasses the engineering necessary to specify a complete bill of materials for the software and the engineering necessary to make significant make/buy trade-offs so that all custom components are elaborated to the extent that construction/assembly costs are highly predictable. Figure 11-4 shows the focus of software architecture team activities over the project life cycle.

For any project, the skill of the software architecture team is crucial. It provides the framework for facilitating team communications, for achieving system-wide qualities, and for implementing the applications. With a good architecture team, an average development team can succeed. If the architecture is weak, even an expert development team of superstar programmers will probably not succeed.

In most projects, the inception and elaboration phases will be dominated by two distinct teams: the software management team and the software architecture team. (Even this distinction may be blurred, depending on scale.) The software development and software assessment teams tend to engage in support roles while preparing for the

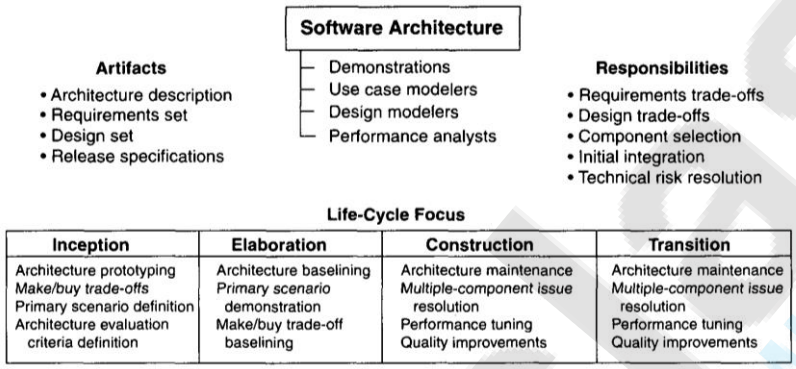
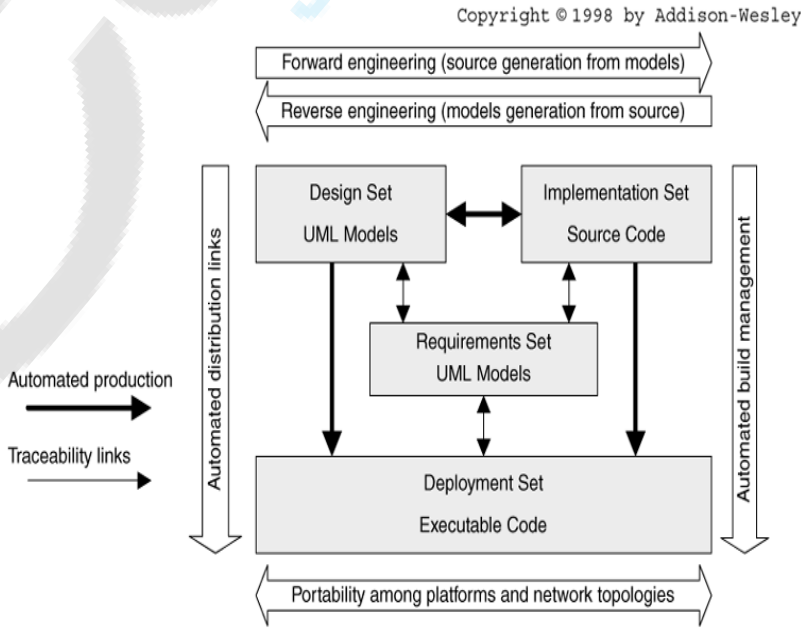


FIGURE 11-4. *Software architecture team activities*

c. Why the process is to be automated? Explain the reasons and the three levels of automation.

SPM---Walker Royce (Pg. No 173 & 174)
Need for automation: (1 mark)
About Round trip Engineering: (1 mark)
Three levels such as
Forward Engineering
Reverse Engineering
Re-Engineering (1 mark)
Diagram: (1 mark) and Explanation of the diagram: (1 mark)



d. Why the tools are considered as the core components in the process automation? Explain.

SPM---Walker Royce (Pg. No. 169)

Many tools are available to automate the software development process. They are considered as the core components in automation process. (1 mark)

Diagram: (1 mark)

Explanation of the tools in brief manner: (6 workflows) (3 marks)

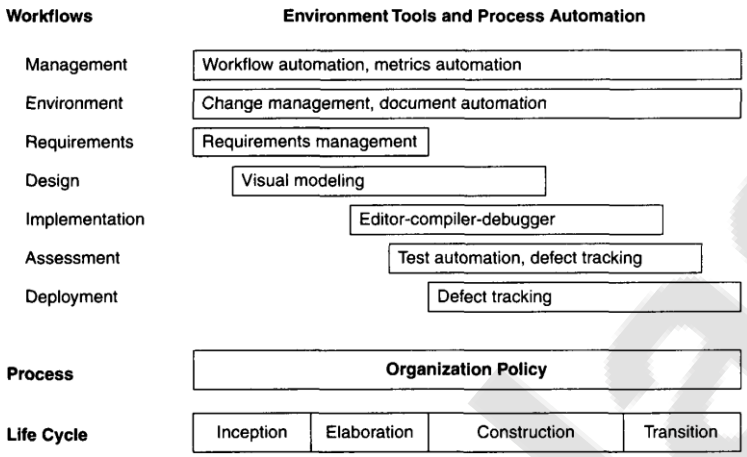


FIGURE 12-1. *Typical automation and tool components that support the process workflows*

work. It introduces some of the important tools that tend to be needed universally across software projects and that correlate well to the process framework. (Many other tools and process automation aids are not included.) Most of the core software development tools map closely to one of the process workflows, as illustrated in Figure 12-1.

Each of the process workflows has a distinct need for automation support. In some cases, it is necessary to generate an artifact; in others, it is needed for simple bookkeeping. Some of the critical concerns associated with each workflow are discussed next.

5. Attempt *any two* of the following: **10**

a. Explain the three management indicators' metrics in detail.

SPM---Walker Royce(Pg No:189---191)

Seven core metrics and its need: (1 mark)

The seven core metricsMany different metrics may be of value in managing a modern process. Seven core metrics that should be used on all software projects. Three are management indicators and four are quality indicators.

Management Indicators (Enlist all the three indicators---1 marks)

1. Work and progress (work performed over time)
2. Budgeted cost and expenditures (cost incurred over time)
3. Staffing and team dynamics (personnel changes over time)

Explanation: (3 indicators---3 marks)

Metric	Purpose	Perspectives
Work and progress	Iteration planning, plan vs. actuals, management indicator	SLOC, function points, object points, scenarios, test cases, SCOs
Budgeted cost and Expenditures	Financial insight, plan vs. actuals, management Indicator	Cost per month, full-time staff per month, percentage of budget expended
Staffing and team dynamics	Resource plan vs. actuals, hiring rate, attrition rate	People per month added, people per month leaving

b. Write a short note on pragmatic software metrics.

SPM---Walker Royce (Pg.No 201)

Any 5 characteristics---5 marks

Basic characteristics of good programmatic software metrics

The basic characteristics of a good metric are as follows:

1) It is **considered meaningfu**l 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 n**atural 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.

c. Enlist the factors of tailoring a software process framework. Explain any of the 4 factors in detail.

SPM---Walker Royce---(Page No: 210---216)

Enlist the factors: (1 mark)

1. **Scale/Size**
2. **Stake holder cohesion**
3. **Flexibility**
4. **Process maturity**
5. **Architecture Risk**
6. **Domain Experience**

Explanation of any 4 points---1 mark each ---(4 marks)

d. Explain about small scale and large scale projects.

SPM---Walker Royce (Pg. No 219-220)

5 points of explanation---5 marks---1 mark each

1. Management is more important in the large projects than in the small projects.
2. Large projects have numerous mundane jobs, especially in the overhead workflows, and more communications.
3. The probability of recruiting, maintaining, and retaining a large number of exceptional people is small.
4. The number of people needed is more important than the project cost for tailoring the project.
5. Size is the most important parameter in the effort formula (generally, lines of source code) and in determining project scale and the number of people needed.
6. Table:

TABLE 14-7. *Schedule distribution across phases for small and large projects*

	ENGINEERING		PRODUCTION	
DOMAIN	INCEPTION	ELABORATION	CONSTRUCTION	TRANSITION
Small commercial project	10%	20%	50%	20%
Large, complex project	15%	30%	40%	15%

6. Attempt *any two* of the following: **10**

a. Enlist the various principles of modern project management.

SPM---Walker Royce(Page No: 231-232)

Enlist the 10 principles and brief explanation of those: (5 marks)

1. Base the process on an **architecture-first** approach. This requires that a demonstrable balance be achieved among the driving requirements, the architecturally significant design decisions, and the life-cycle plans before the resources are committed for full-scale development.

2) Establish an **iterative life-cycle** process that confronts risk early. With today's sophisticated software systems, it is not possible to define the entire problem, design the entire solution, build the software, and then test the end product in sequence. Instead, an iterative process that refines the problem understanding, an effective solution, and an effective plan over several iterations encourages a balanced treatment of all stakeholder objectives. Major risks must be addressed early to increase predictability and avoid expensive downstream scrap and rework.

3. Transition design methods to emphasize **component-based** development. Moving from a line-of-code mentality to a component-based mentality is necessary to reduce the amount of human-generated source code and custom development. A component is a cohesive set of preexisting lines of code, either in source or executable format, with a defined interface
and behavior.
4) Establish a **change management** environment. The dynamics of iterative development, including concurrent workflows by different teams working on shared artifacts, necessitates objectively controlled baselines.
5) Enhance change freedom though tools that support round trip engineering**.**
Round trip engineering is the environment support necessary to automate and synchronize engineering information in different formats (such as requirements specifications, design models, source code, executable code, test cases). Without substantial automation of this bookkeeping, change management, documentation, and testing, it is difficult to reduce iteration cycles to manageable time frames in which change is encouraged rather than avoided. Change freedom is a necessity in an iterative process, and establishing an integrated environment is crucial.
6) Capture design artifacts in rigorous, **model-based notation**. A model- based approach (such as UML) supports the evolution of semantically rich graphical and textual design notations. Visual modeling with rigorous notations and a formal machine-processable language provides for far more objective measures than the traditional approach of human review and inspection of ad hoc design representations in paper documents.
7. **Objective quality control**
8. Demo based approach
9. Evolving levels of details
10. Establish a configurable process

b. Explain the best practices listed it out by "Airline Software Council".
SPM---walker Royce (Pg No: 233,234)
9 principles enlist-along with brief explanation---5 marks
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 baseline 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) Program-wide visibility of progress versus plan.

This practice—namely, open communications among project team Members—is obviously necessary.

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.

c. Analyze the future perspective of cost estimation model.

Future Perspective or the next generation cost estimation model:

SPM---Walker Royce (Page No: 238,239)

About future cost model : (1 mark)

Software experts hold widely varying opinions about software economics and its manifestation in software cost estimation models:

Factors to be considered with explanation : (2 marks)

- **SLOC Vs. FP**
- **Productivity Vs. Quality Measures**
- **Java Vs. CPP**
- **Object-oriented Vs. Functionally oriented**

It will be difficult to improve empirical estimation models while the project data going into these models are noisy and highly uncorrelated, and are based on differing process and technology foundations.

Some of today's popular software cost models are not well matched to an iterative software process focused an architecture-first approach.

Many cost estimators are still using a conventional process experience base to estimate a modern project profile.

A next-generation software cost model should explicitly separate architectural engineering from application production, just as an architecture-first process does.

Two major improvements in next-generation software cost estimation models: (2 marks)

1. **Separation of the engineering stage from the production stage** will force estimators to differentiate between architectural scale and implementation size.
2. **Rigorous design notations** such as UML will offer an opportunity to define units of measure for scale that are more standardized and therefore can be automated and tracked.

d. Write a short note on the process of denouement.

SPM---Walker Royce (Page No: 252,253)

Introduction: (1 mark)

Good way to transition to a more mature iterative development process that supports automation technologies and modern architectures is to take the following shot:

Ready Aim and fire (3 marks)
Ready.
Do your homework. Analyze modern approaches and technologies. Define your process. Support it with mature environments, tools, and components. Plan thoroughly.
Aim:
Select a critical project. Staff it with the right team of complementary resources and demand improved results.
Fire:
Execute the organizational and project-level plans with vigor and follow-through.
Conclusion: (1 mark)

7. Attempt *any three* of the following: **15**

a. "Late design & integration is one of the biggest drawbacks of waterfall model"- Explain the statement.
SPM---Walker Royce (Page No: 12 & 13)
Explanation of the statement: 1 mark
Diagram 1 mark
and Explanation: 3 marks

Protracted Integration and Late Design Breakage

For a typical development project that used a waterfall model management process, Figure 1-2 illustrates development progress versus time. Progress is defined as percent coded, that is, demonstrable in its target form. (The software was compilable and executable; it was not necessarily complete, compliant, nor up to specifications.) The following sequence was common:

- Early success via paper designs and thorough (often *too* thorough) briefings
- Commitment to code late in the life cycle
- Integration nightmares due to unforeseen implementation issues and interface ambiguities
- Heavy budget and schedule pressure to get the system working
- Late shoe-horning of nonoptimal fixes, with no time for redesign
- A very fragile, unmaintainable product delivered late

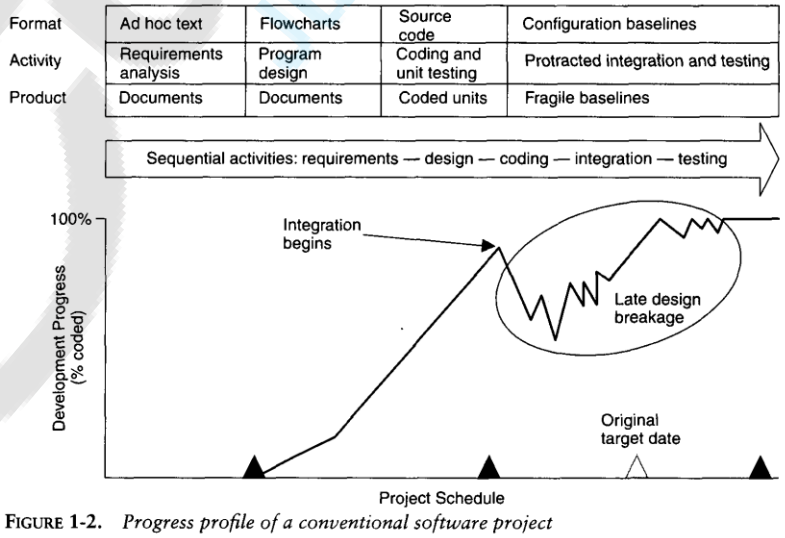


FIGURE 1-2. *Progress profile of a conventional software project*

b. Write a short note on pragmatic artifacts.

SPM---Walker Royce (Page No:105---107)
What is pragmatic artifacts? (1 mark)
Philosophy that arises issues: (any 4 points with brief explanation---4 marks)

1. **People want to review the information but don't understand the language of the artifacts**
2. **People want to review the information but don't have the access to tools**
3. **Artifacts should use rigorous notations**
4. **Useful documentation is self-defining**
5. **Paper is tangible; e-documents are easy to change**

c. Explain the 2 perspectives through which project plans need to be derived.

SPM---Walker Royce (Page No: 140-143)
Project plans are derived through (1 mark)
Conventional WBS
Evolutionary WBS
Explanation of both (4 marks---2 marks each)

d. Explain the nature of project-team members and their profiles.

SPM---Walker Royce (Page No: 155 & 156)
Project Team Members: (5 members and their roles in brief---5 marks---1 mark each)
Project Manager
Team Leader
Team Members
Customers
Stake holders
Above mentioned answers can be written like

1. **SEPA, (1 mark)**
2. **SEEA, (1 mark)**
3. **Project Review Authority (1 mark)**

and

4. **Infrastructure (1 mark)**

Diagram to show the roles (1 mark)

e. Explain the two dimensions of process discriminate with neat diagram.

SPM---Walker Royce (Page No: 209,210)

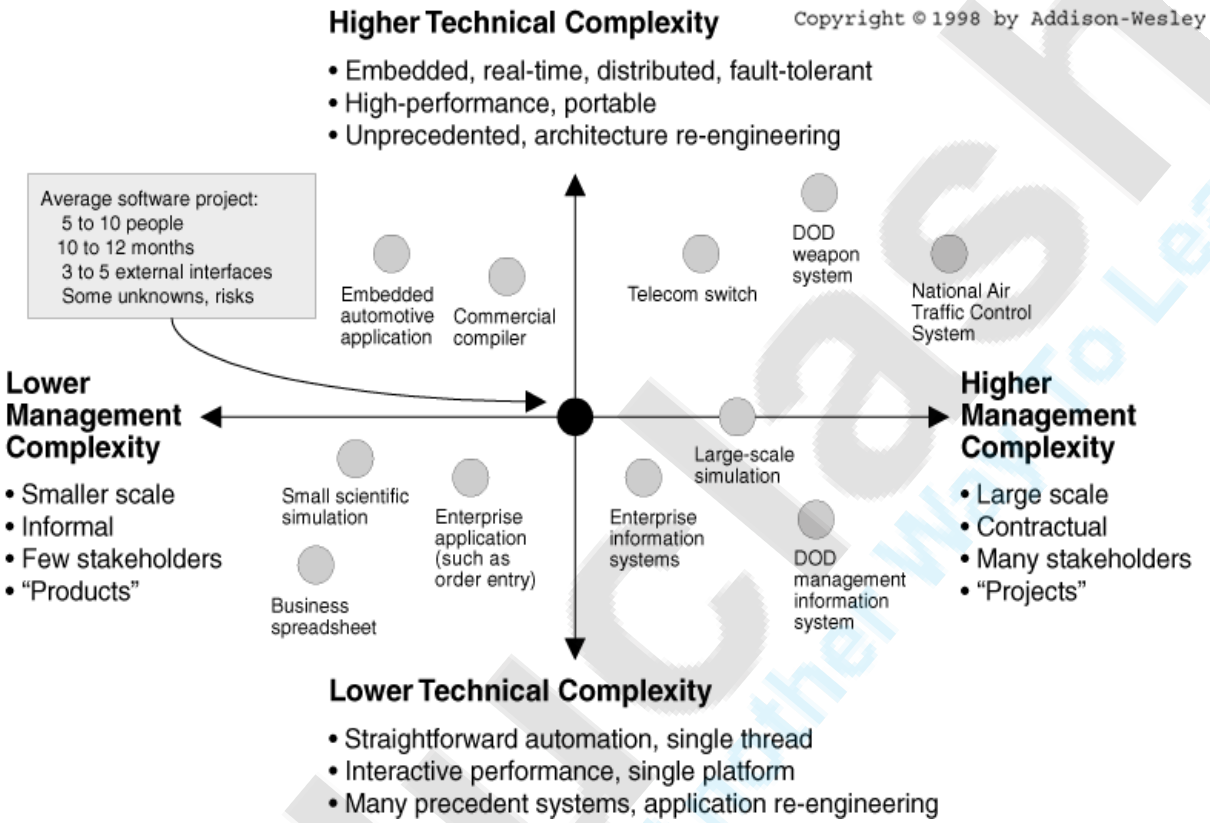
Enlist the two dimensions : (1 mark)

1. **Technical and**
2. **management complexity**

Brief explanation about these two dimensions: (1 mark)

Diagram (1 mark)

Explanation of the diagram (2 marks)



f. Explain the basic difference between problem space Vs. Solution space with respect to software cost model.

SPM---Walker Royce (Page No: 240)
Next level of automation point (1 mark)
Re usability point (1 mark)
Architectures Vs. Application composition point (1 mark)
Diagram (1 mark)
and
Explanation of the diagram (1 mark)

and standardize many of these management activities, thereby requiring a lower percentage of effort for overhead activities as scale increases.

Reusing common processes across multiple iterations of a single project, multiple releases of a single product, or multiple projects in an organization also relieves many of the sources of diseconomy of scale. Critical sources of scrap and rework are eliminated by applying precedent experience and mature processes. Establishing trustworthy plans based on credible project performance norms and using reliable components reduce other sources of scrap and rework. While most reuse of components results in reducing the size of the production effort, the reuse of processes, tools, and experience has a direct impact on the economies of scale.

Another important difference in this cost model is that architectures and applications have different units of mass (scale versus size) and are representations of the solution space. Scale might be measured in terms of architecturally significant elements (classes, components, processes, nodes), and size might be measured in SLOC or megabytes of executable code. These measures differ from measures of the problem space such as discrete requirements or use cases. The problem space description certainly drives the definition of the solution space. However, there are many solutions to any given problem, as illustrated in Figure 16-2, each with a different value proposition. Cost is a key discriminator among potential solutions. Cost estimates that are more accurate and more precise can be derived from specific solutions to problems. Therefore, the cost estimation model must be governed by the basic parameters of a

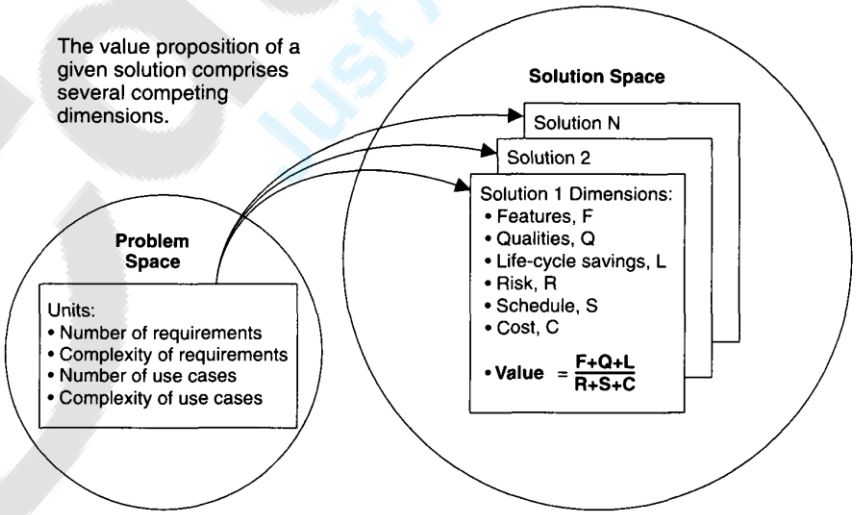


FIGURE 16-2. *Differentiating potential solutions through cost estimation*