UNIT-05/LECTURE-01

Matrix Organization: [RGPV/June 2012(10)]

An organizational structure that facilitates the horizontal flow of skills and information. It is used mainly in the management of large projects or product development processes, drawing employees from different functional disciplines for assignment to a team without removing them from their respective positions.

Employees in a matrix organization report on day-to-day performance to the project or product manager whose authority flows sideways (horizontally) across departmental boundaries. They also continue to report on their overall performance to the head of their department whose authority flows downwards (vertically) within his or her department. In addition to a multiple command and control structure, a matrix organization necessitates new support mechanisms, organizational culture, and behaviour patterns. Developed at the US National Aeronautics & Space Administration (NASA) in association with its suppliers, this structure gets its name from its resemblance to a table (matrix) where every element is included in a row as well as a column.

RMMM[RGPV/June 2012(10)]

RISK MITIGATION, MONITORING, AND MANAGEMENT (RMMM) PLAN

The RMMM Plan maybe developed in the form of a document. Alternatively, an organization may create a set of Risk Information Sheets (RIS) [often in electronic form] that contain all pertinent information outlined below. IMPORTANT note: Like most software engineering documents, the RMMM Plan evolves over time.

1.0 Introduction

This section provides an overview of the RMMM Plan.

1.1 Scope and intent of RMMM activities

A description of the overall RMMM focus including objectives, organizational responsibilities.

1.2 Risk management organizational role

Description of who has responsibility for risk management.

2.0 Project risks

This section describes all known project risks.

2.1 Risk table

A presentation of all risks, probabilities and impact.

2.1.1 Description of risk *m*

Risk m is described along with relevant sub conditions. Section2.1.1 is repeated for each of m risks.

2.1.2 Probability and impact for risk *m*

Probability and impact for risk m is described Section 2.1.2 is repeated for each of m risks.

2.2 Risk refinement

High probability/high impact risks are refined using the CTC approach.

3.0 Risk mitigation, monitoring, and management

This section discusses RMMM for each risk.

3.1 Risk mitigation for risk *m*

How do we avoid risk m? Section 3.1 is repeated for each of m risks.

3.2 Risk monitoring for risk *m*

What conditions do we monitor to determine whether risk m is becoming more or less likely? Section3.2 is repeated for each of m risks.

3.3 Risk management for risk *m*

What contingency plans should we put into plan under the assumption that risk m will occur? Section 3.3 is repeated for each of m risks.

Software Project Scheduling[RGPV/June 2012(10),June 2014(5)]

- Introduction
- Project scheduling
- Task network
- Timeline chart
- Earned value analysis

Introduction

Eight Reasons for Late Software Delivery

- An unrealistic deadline established by someone outside the software engineering group and forced on managers and practitioners within the group
- Changing customer requirements that are not reflected in schedule changes
- An honest underestimate of the amount of effort and /or the number of resources that

will be required to do the job

- Predictable and/or unpredictable risks that were not considered when the project commenced
- Technical difficulties that could not have been foreseen in advance
- Human difficulties that could not have been foreseen in advance
- Miscommunication among project staff that results in delays
- A failure by project management to recognize that the project is falling behind schedule and a lack of action to correct the problem

Handling Unrealistic Deadlines

- Perform a detailed estimate using historical data from past projects; determine the estimated effort and duration for the project
- Using an incremental model, develop a software engineering strategy that will deliver critical functionality by the imposed deadline, but delay other functionality until later; document the plan
- Meet with the customer and (using the detailed estimate) explain why the imposed deadline is unrealistic
  - Be certain to note that all estimates are based on performance on past projects
  - Also be certain to indicate the percent improvement that would be required to achieve the deadline as it currently exists

1) Offer the incremental development strategy as an alternative and offer some options
   - Increase the budget and bring on additional resources to try to finish sooner
   - Remove many of the software functions and capabilities that were requested
   - Dispense with reality and wish the project complete using the prescribed schedule; then point out that project history and your estimates show that this is unrealistic and will result in a disaster

Project Scheduling

General Practices

- On large projects, hundreds of small tasks must occur to accomplish a larger goal
  - Some of these tasks lie outside the mainstream and may be completed without worry of impacting on the project completion date
  - Other tasks lie on the critical path; if these tasks fall behind schedule, the completion date of the entire project is put into jeopardy
- Project manager's objectives

  - Define all project tasks
  - Build an activity network that depicts their interdependencies
  - Identify the tasks that are critical within the activity network
  - Build a timeline depicting the planned and actual progress of each task.
  - Track task progress to ensure that delay is recognized “one day at a time”.
  - To do this, the schedule should allow progress to be monitored and the project to be controlled.
- On large projects, hundreds of small tasks must occur to accomplish a larger goal.
  - Some of these tasks lie outside the mainstream and may be completed without worry of impacting on the project completion date
  - Other tasks lie on the critical path; if these tasks fall behind schedule, the completion date of the entire project is put into jeopardy.
- Project manager’s objectives
  - Define all project tasks
  - Build an activity network that depicts their interdependencies.
  - Identify the tasks that are critical within the activity network.
  - Build a timeline depicting the planned and actual progress of each task.
  - Track task progress to ensure that delay is recognized “one day at a time”.
  - To do this, the schedule should allow progress to be monitored and the project to be controlled.

Basic Principles for Project Scheduling

- Compartmentalization
  - The project must be compartmentalized into a number of manageable activities, actions, and tasks; both the product and the process are decomposed.
- Interdependency
  - The interdependency of each compartmentalized activity, action, or task must be determined.
  - Some tasks must occur in sequence while others can occur in parallel.
  - Some actions or activities cannot commence until the work product produced by another is available.
- Time allocation
  - Each task to be scheduled must be allocated some number of work units.
  - In addition, each task must be assigned a start date and a completion date that is

a function of the interdependencies.

  - Start and stop dates are also established based on whether work will be conducted on a full-time or part-time basis.

- Effort validation
  - Every project has a defined number of people on the team.
  - As time allocation occurs, the project manager must ensure that no more than the allocated number of people has been scheduled at any given time.
- Defined responsibilities
  - Every task that is scheduled should be assigned to a specific team member.
- Defined outcomes
  - Every task that is scheduled should have a defined outcome for software projects such as a work product or part of a work product.
  - Work products are often combined in deliverables.
- Defined milestones
  - Every task or group of tasks should be associated with a project milestone
  - A milestone is accomplished when one or more work products has been reviewed for quality and has been approved

S.NO	RGPV QUESTION	YEAR	MARKS
Q.1	Briefly describe Software project management standards.	Jun.2012	10
Q.2	Write short notes on Software project management standards.	Jun.2014	5
Q.3	Explain Matrix Organization in detail.	June2011	10
Q.4	Explain RMMM in detail	June2012	10

UNIT-05/LECTURE-02

Task Network:

Defining a Task Set

- A task set is the work breakdown structure for the project
- No single task set is appropriate for all projects and process models
  - It varies depending on the project type and the degree of rigor (based on influential factors) with which the team plans to work
- The task set should provide enough discipline to achieve high software quality
  - But it must not burden the project team with unnecessary work

Types of Software Projects

- Concept development projects
  - Explore some new business concept or application of some new technology
- New application development
  - Undertaken as a consequence of a specific customer request
- Application enhancement
  - Occur when existing software undergoes major modifications to function, performance, or interfaces that are observable by the end user
- Application maintenance
  - Correct, adapt, or extend existing software in ways that may not be immediately obvious to the end user
- Reengineering projects
  - Undertaken with the intent of rebuilding an existing (legacy) system in whole or in part

Factors that Influence a Project's Schedule

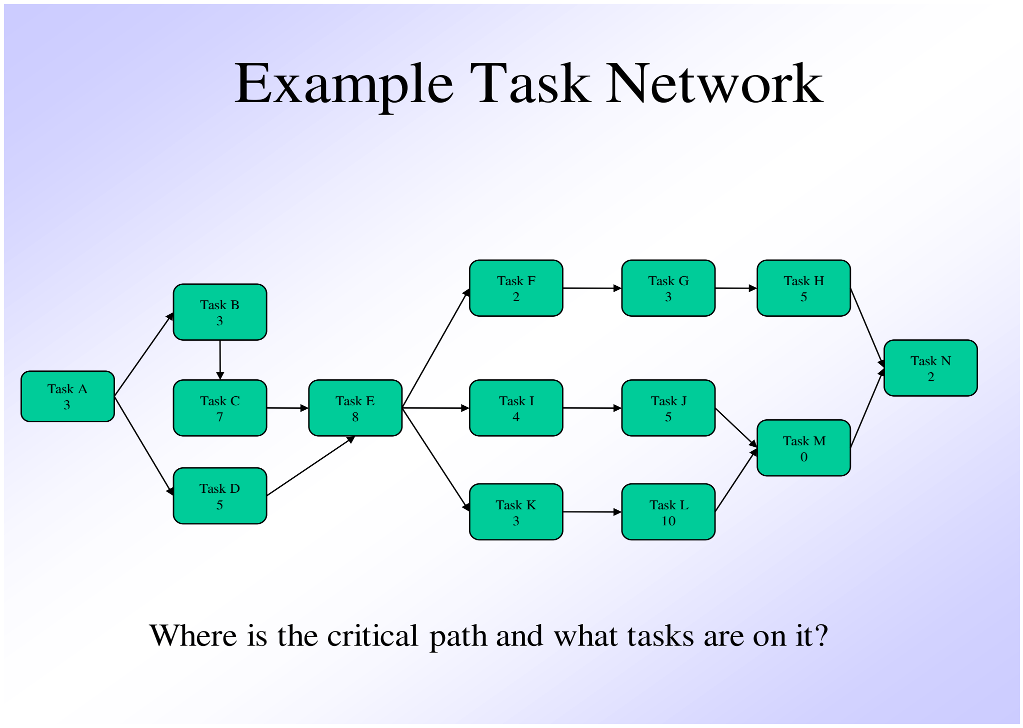
- Size of the project
- Number of potential users
- Mission criticality
- Application longevity
- Stability of requirements
- Ease of customer/developer communication
- Maturity of applicable technology

we dont take any liability for the notes correctness. http://www.rgpvonline.com

- Performance constraints
- Embedded and non-embedded characteristics
- Project staff
- Reengineering factors

Purpose of a Task Network

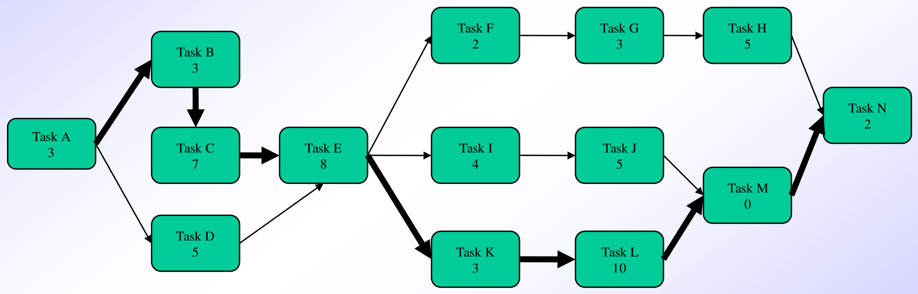
- Also called an activity network
- It is a graphic representation of the task flow for a project
- It depicts task length, sequence, concurrency, and dependency
- Points out inter-task dependencies to help the manager ensure continuous progress toward project completion
- The critical path
  - A single path leading from start to finish in a task network
  - It contains the sequence of tasks that must be completed on schedule if the project as a whole is to be completed on schedule
  - It also determines the minimum duration of the project



Example Task Network

Where is the critical path and what tasks are on it?

Example Task Network with Critical Path Marked



Critical path: A-B-C-E-K-L-M-N

UNIT-05/LECTURE-03

Timeline Chart

Mechanics of a Timeline Chart

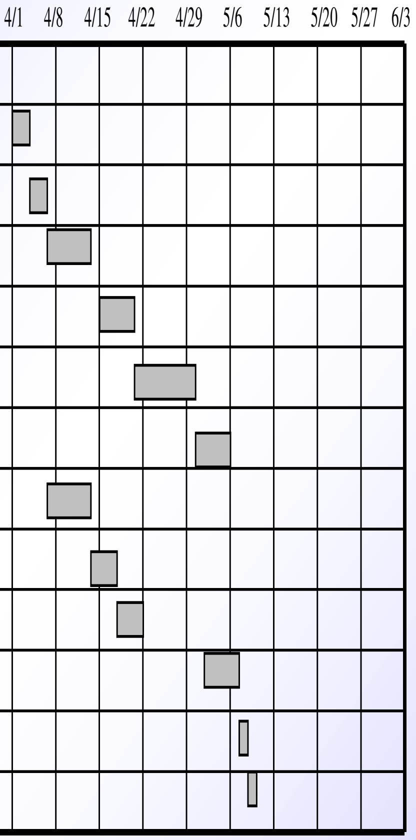
- Also called a Gantt chart; invented by Henry Gantt, industrial engineer, 1917
- All project tasks are listed in the far left column
- The next few columns may list the following for each task: projected start date, projected stop date, projected duration, actual start date, actual stop date, actual duration, task inter-dependencies (i.e., predecessors)
- To the far right are columns representing dates on a calendar
- The length of a horizontal bar on the calendar indicates the duration of the task
- When multiple bars occur at the same time interval on the calendar, this implies task concurrency
- A diamond in the calendar area of a specific task indicates that the task is a milestone; a milestone has a time duration of zero

we dont take any liability for the notes correctness. http://www.rgpvonline.com

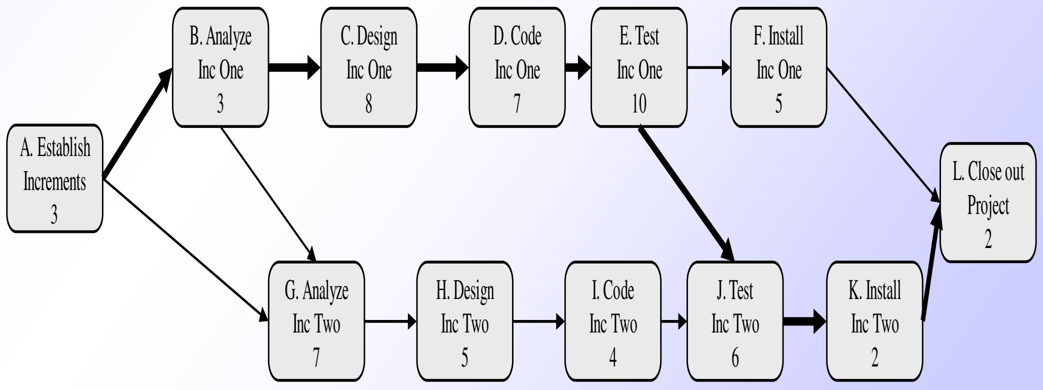
SOLUTION

Timeline chart:

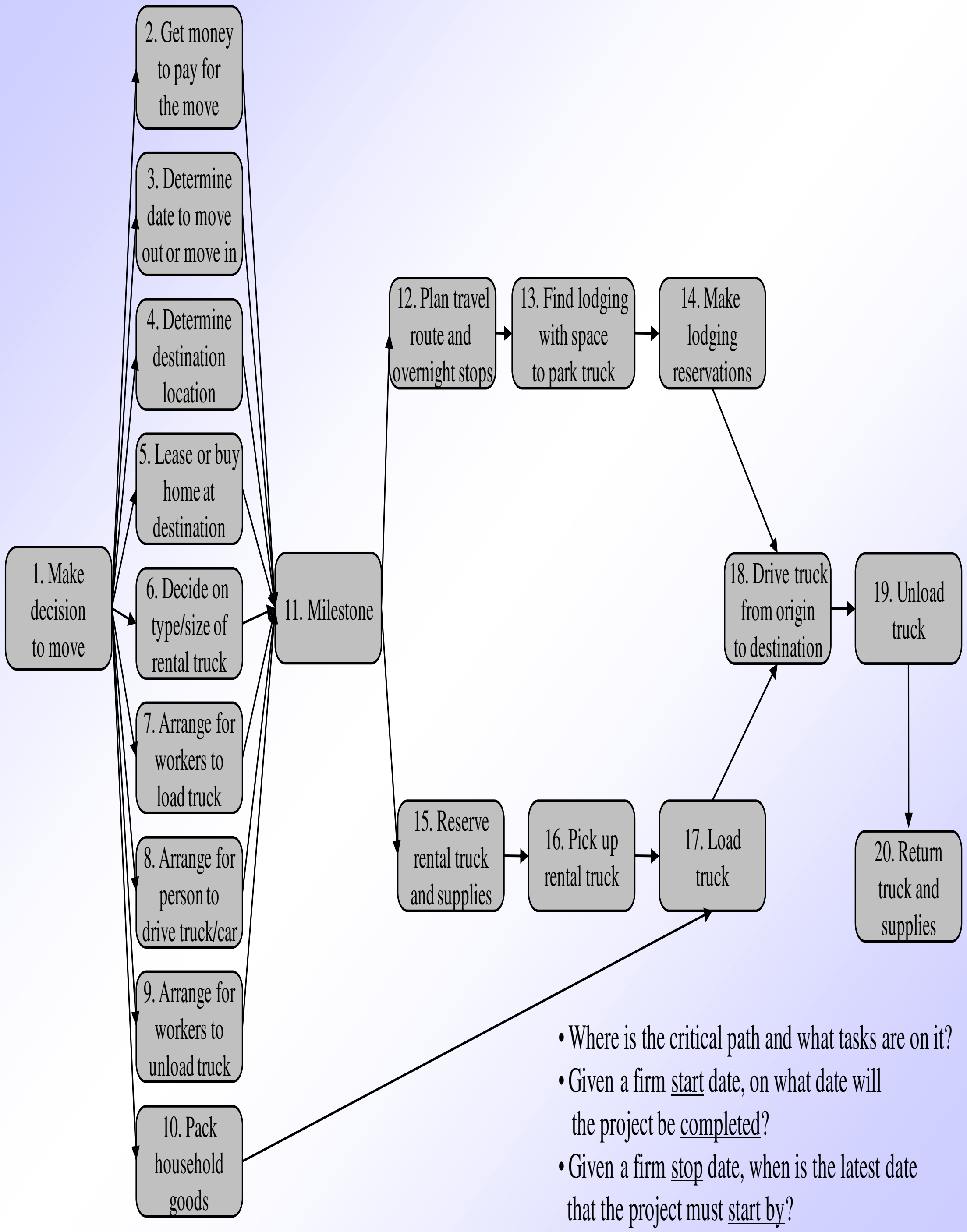
Task #	Task Name	Duration	Start	Finish	Pred.
A	Establish increments	3	4/1	4/3	None
B	Analyze Inc One	3	4/4	4/6	A
C	Design Inc One	8	4/7	4/14	B
D	Code Inc One	7	4/15	4/21	C
E	Test Inc One	10	4/22	5/1	D
F	Install Inc One	5	5/2	5/6	E
G	Analyze Inc Two	7	4/7	4/13	A, B
H	Design Inc Two	5	4/14	4/18	G
I	Code Inc Two	4	4/19	4/22	H
J	Test Inc Two	6	5/2	5/7	E, I
K	Install Inc Two	2	5/8	5/9	J
L	Close out project	2	5/10	5/11	F, K



Task network and the critical path: A-B-C-D-E-J-K-L

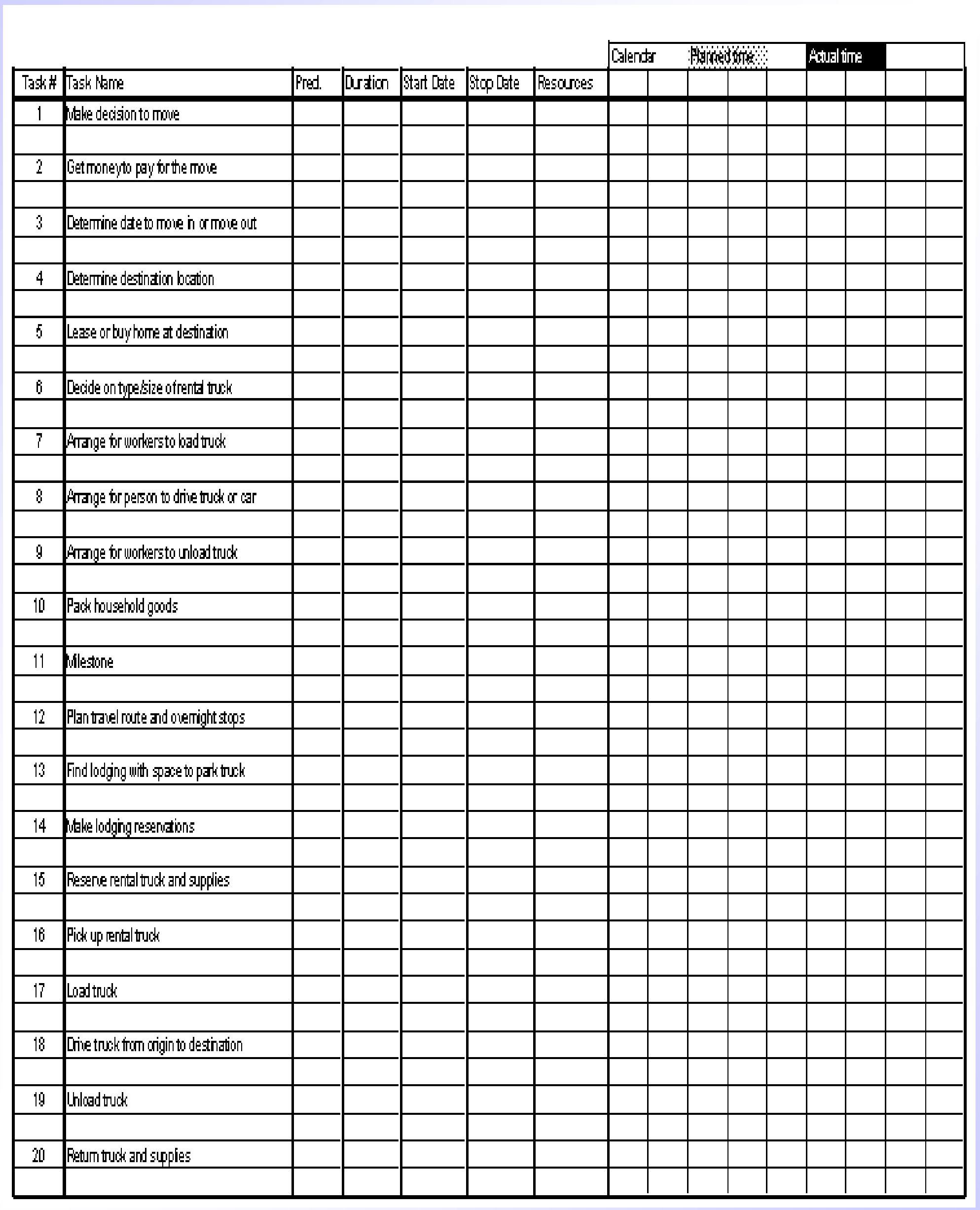


Task Network for a Long-Distance Move of 8,000 lbs of Household Goods


1. Make decision to move
2. Get money to pay for the move
3. Determine date to move out or move in
4. Determine destination location
5. Lease or buy home at destination
6. Decide on type/size of rental truck
7. Arrange for workers to load truck
8. Arrange for person to drive truck/car
9. Arrange for workers to unload truck
10. Pack household goods
11. Milestone
12. Plan travel route and overnight stops
13. Find lodging with space to park truck
14. Make lodging reservations
15. Reserve rental truck and supplies
16. Pick up rental truck
17. Load truck
18. Drive truck from origin to destination
19. Unload truck
20. Return truck and supplies

- Where is the critical path and what tasks are on it?
- Given a firm start date, on what date will the project be completed?
- Given a firm stop date, when is the latest date that the project must start by?

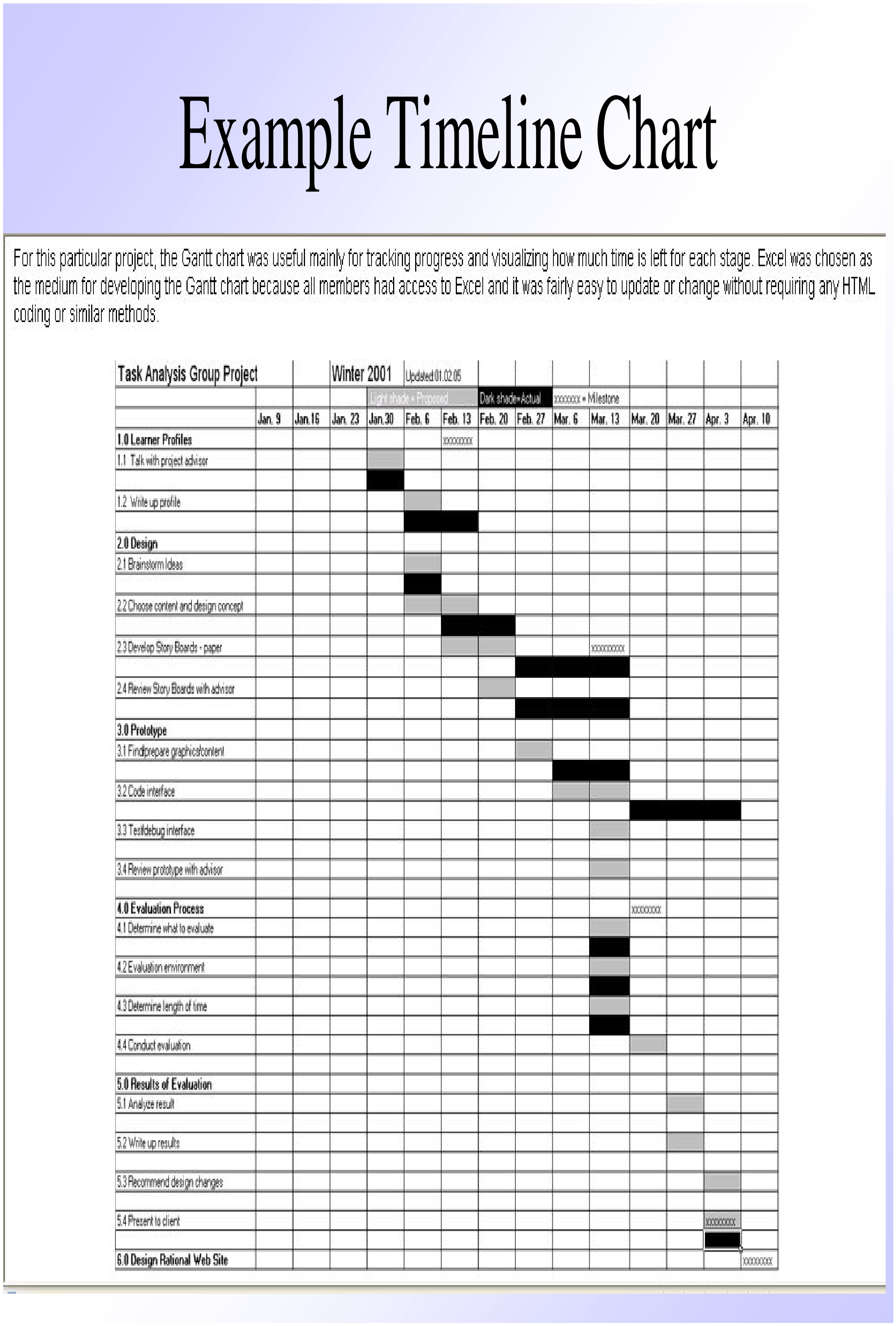
Timeline Chart for Long Distance Move



							Calendar		Planned time			Actual time			
Task #	Task Name	Pred.	Duration	Start Date	Stop Date	Resources									
1	Make decision to move														
2	Get money to pay for the move														
3	Determine date to move in or move out														
4	Determine destination location														
5	Lease or buy home at destination														
6	Decide on type/size of rental truck														
7	Arrange for workers to load truck														
8	Arrange for person to drive truck or car														
9	Arrange for workers to unload truck														
10	Pack household goods														
11	Milestone														
12	Plan travel route and overnight stops														
13	Find lodging with space to park truck														
14	Make lodging reservations														
15	Reserve rental truck and supplies														
16	Pick up rental truck														
17	Load truck														
18	Drive truck from origin to destination														
19	Unload truck														
20	Return truck and supplies														

UNIT-05/LECTURE-04

Example Timeline Chart:



Example Timeline Chart

For this particular project, the Gantt chart was useful mainly for tracking progress and visualizing how much time is left for each stage. Excel was chosen as the medium for developing the Gantt chart because all members had access to Excel and it was fairly easy to update or change without requiring any HTML coding or similar methods.

Task Analysis Group Project — Winter 2001 — Updated 01.02.05

Light shade = Proposed ▨ — Dark shade=Actual ■ — xxxxxxx = Milestone

	Jan. 9	Jan.16	Jan. 23	Jan.30	Feb. 6	Feb. 13	Feb. 20	Feb. 27	Mar. 6	Mar. 13	Mar. 20	Mar. 27	Apr. 3	Apr. 10
1.0 Learner Profiles						xxxxxxxx								
1.1 Talk with project advisor				▨										
				■										
1.2 Write up profile					▨									
					■	■								
2.0 Design														
2.1 Brainstorm Ideas					▨									
					■									
2.2 Choose content and design concept					▨	▨								
						■	■							
2.3 Develop Story Boards - paper						▨	▨			xxxxxxxxx				
								■	■	■				
2.4 Review Story Boards with advisor							▨							
								■	■	■				
3.0 Prototype														
3.1 Find/prepare graphics/content								▨						
									■	■				
3.2 Code interface									▨	▨				
											■	■	■	
3.3 Test/debug interface										▨				
3.4 Review prototype with advisor										▨				
4.0 Evaluation Process											xxxxxxxx			
4.1 Determine what to evaluate										▨				
										■				
4.2 Evaluation environment										▨				
										■				
4.3 Determine length of time										▨				
										■				
4.4 Conduct evaluation											▨			
5.0 Results of Evaluation														
5.1 Analyze result												▨		
5.2 Write up results												▨		
5.3 Recommend design changes													▨	
5.4 Present to client													xxxxxxxx	
													■	
6.0 Design Rational Web Site														xxxxxxxx

Methods for Tracking the Schedule

- **Qualitative approaches**
  - Conduct periodic project status meetings in which each team member reports progress and problems
  - Evaluate the results of all reviews conducted throughout the software engineering process
  - Determine whether formal project milestones (i.e., diamonds) have been accomplished by the scheduled date
  - Compare actual start date to planned start date for each project task listed in the timeline chart
  - Meet informally with the software engineering team to obtain their subjective assessment of progress to date and problems on the horizon
- **Quantitative approach**
  - Use earned value analysis to assess progress quantitatively

Project Control and Time Boxing

- The project manager applies control to administer project resources, cope with problems, and direct project staff
- If things are going well (i.e., schedule, budget, progress, milestones) then control should be light
- When problems occur, the project manager must apply tight control to reconcile the problems as quickly as possible. For example:
  - Staff may be redeployed
  - The project schedule may be redefined
- Severe deadline pressure may require the use of time boxing
  - An incremental software process is applied to the project
  - The tasks associated with each increment are “time-boxed” (i.e., given a specific start and stop time) by working backward from the delivery date
  - The project is not allowed to get “stuck” on a task
  - When the work on a task hits the stop time of its box, then work ceases on that task and the next task begins
  - This approach succeeds based on the premise that when the time-box boundary

is encountered, it is likely that 90% of the work is complete

  - The remaining 10% of the work can be
    - Delayed until the next increment

Milestones for OO Projects

- Task parallelism in object-oriented projects makes project tracking more difficult to do than non-OO projects because a number of different activities can be happening at once
- Sample milestones
  - Object-oriented analysis completed
  - Object-oriented design completed
  - Object-oriented coding completed
  - Object-oriented testing completed
- Because the object-oriented process is an iterative process, each of these milestones may be revisited as different increments are delivered to the customer

Earned Value Analysis

- Earned value analysis is a measure of progress by assessing the percent of completeness for a project
- It gives accurate and reliable readings of performance very early into a project
- It provides a common value scale (i.e., time) for every project task, regardless of the type of work being performed
- The total hours to do the whole project are estimated, and every task is given an earned value based on its estimated percentage of the total

Determining Earned Value

- Compute the budgeted cost of work scheduled (BCWS) for each work task *i* in the schedule
  - The BCWS is the effort planned; work is estimated in person-hours or person-days for each task
  - To determine progress at a given point along the project schedule, the value of BCWS is the sum of the $BCWS_i$ values of all the work tasks that should have been completed by that point of time in the project schedule
- Sum up the BCWS values for all work tasks to derive the budget at completion (BAC)

- Compute the value for the budgeted cost of work performed (BCWP)
  - BCWP is the sum of the BCWS values for all work tasks that have actually been completed by a point of time on the project schedule

Progress Indicators provided through Earned Value Analysis

- SPI = BCWP/BCWS
  - Schedule performance index (SPI) is an indication of the efficiency with which the project is utilizing scheduled resources
  - SPI close to 1.0 indicates efficient execution of the project schedule
- SV = BCWP – BCWS
  - Schedule variance (SV) is an absolute indication of variance from the planned schedule
- PSFC = BCWS/BAC
  - Percent scheduled for completion (PSFC) provides an indication of the percentage of work that should have been completed by time *t*
- PC = BCWP/BAC
  - Percent complete (PC) provides a quantitative indication of the percent of work that has been completed at a given point in time *t*
- ACWP = sum of BCWP as of time t
  - Actual cost of work performed (ASWP) includes all tasks that have been completed by a point in time *t* on the project schedule
- CPI = BCWP/ACWP
  - A cost performance index (CPI) close to 1.0 provides a strong indication that the project is within its defined budget
- CV = BCWP – ACWP
  - The cost variance is an absolute indication of cost savings (against planned costs) or shortfall at a particular stage of a project

UNIT-04/LECTURE-05

Re-Engineering

Restructuring or rewriting part or all of a system without changing its functionality

- Applicable when some (but not all) subsystems of a larger system require frequent maintenance
- Reengineering involves putting in the effort to make it easier to maintain
- The reengineered system may also be restructured and should be redocumented

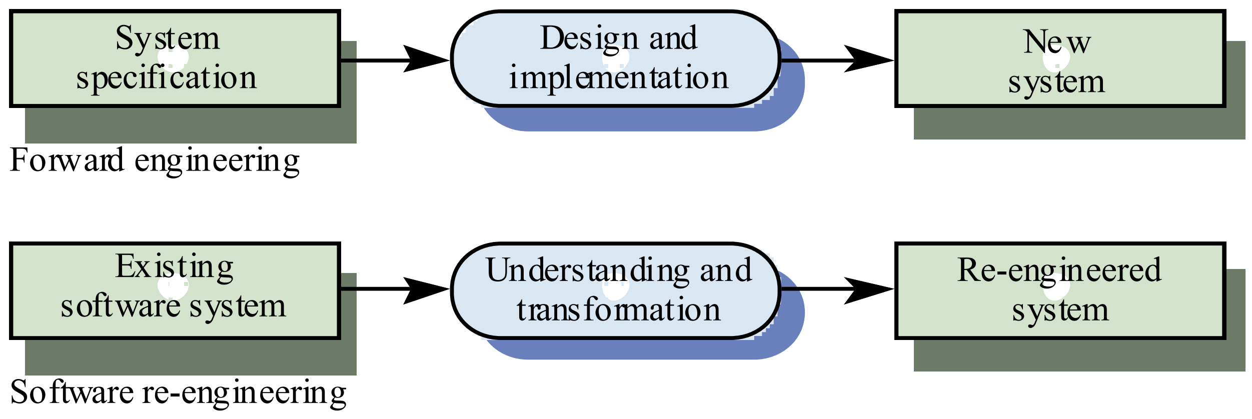
When do you decide to reengineer?

- When system changes are confined to one subsystem, the subsystem needs to be reengineered
- When hardware or software support becomes obsolete
- When tools to support restructuring are readily available

Business Process Reengineering

- Concerned with redesigning business processes to make them more responsive and more efficient
- Often relies on the introduction of new computer systems to support the revised processes
- May force software reengineering of legacy computer systems which designed to support existing processes

Reengineering Process



- **Inventory analysis**
  - sorting active software applications by business criticality, longevity, current maintainability, and other local criteria
  - helps to identify reengineering candidates

- **Document restructuring options**
  - live with weak documentation
  - update poor documents if they are used
  - fully rewrite the documentation for critical systems focusing on the "essential minimum"
- **Reverse engineering**
  - process of design recovery
  - analyzing a program in an effort to create a representation of the program at some abstraction level higher than source code
- **Code restructuring**
  - source code is analyzed and violations of structured programming practices are noted and repaired
  - revised code needs to be reviewed and tested
- **Data restructuring**
  - usually requires full reverse engineering
  - current data architecture is dissected
  - data models are defined
  - existing data structures are reviewed for quality
- **Forward engineering**
  - sometimes called reclamation or renovation
  - recovers design information from existing source code
  - uses this design information to reconstitute the existing system to improve its overall quality or performance

UNIT-05/LECTURE-06

Forward Engineering[RGPV/June 2014(5)]

Forward engineering is the process of building from a high-level model or concept to build in complexities and lower-level details. This type of engineering has different principles in various software and database processes.

Generally, forward engineering is important in IT because it represents the 'normal' development process. For example, building from a model into an implementation language. This will often result in loss of semantics, if models are more semantically detailed, or levels of abstraction.

Forward engineering is thus related to the term 'reverse engineering,' where there is an effort to build backward, from a coded set to a model, or to unravel the process of how something was put together.

Forward Engineering Client/Server Architectures

- application functionality migrates to each client computer
- new GUI interfaces implemented at client sites
- database functions allocated to servers
- specialized functionality may remain at server site
- new communications, security, archiving, and control requirements must be established at both client and server sites

Forward Engineering User Interfaces

- understand the original user interface and how the data moves between the user interface and the remainder of the application
- remodel the behavior implied by the existing user interface into a series of abstractions that have meaning in the context of a GUI
- introduce improvements that make the mode of interaction more efficient build and integrate the new GUI

Web Engineering[RGPV/June 2013(7)]

- There is still no common answer

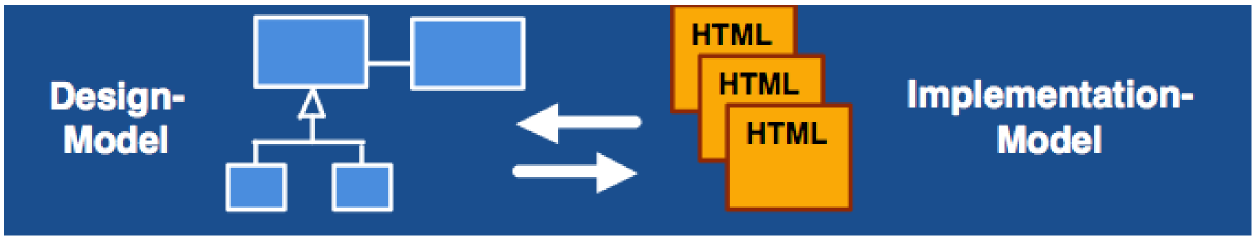
  - “Web Engineering is concerned with establishment and use of sound scientific, engineering and management principles and disciplined and systematic approaches to the successful development, deployment and maintenance of high quality Web-based systems and applications”, SIGWEB Newsletter
  - “Web Engineering is a discipline among disciplines, cutting across computer science, information systems, and software engineering, as well as benefiting from several non-IT specializations”, IEEE Multimedia
  - “While Web Engineering adopts and encompasses many software engineering principles, it incorporates many new approaches, methodologies, tools, techniques, and guidelines to meet the unique requirements of Web-based systems. Developing Web-based systems is significantly different from traditional software development and poses many additional challenges”, IEEE MultiMedia
  - The application of systematic, disciplined, and quantifiable approaches to the design, production, deployment, operation, maintenance and evolution of Web-based software products.

- A holistic and proactive approach to Web systems development
- Offers systematic approaches and disciplined processes for development
- Deals with the management of complexity and diversity of Web development
- Brings to Web-based system development
  - Control
  - Risk minimization
  - Enhanced maintainability and quality
- Other factors
  - Document orientation
  - Navigational design
  - Changing technology
  - Budget and time constraints
  - People and internal politics
  - Division between theory and practice
  - Lack of understanding

- Web System – an infrastructure or system enabling the operation of a Web application
- Web Application – a distributed application that accomplishes a certain business need

- based on the technologies of WWW and that consists of a set of Web-specific resources

Web Application Development



- Still Ad-Hoc instead of a disciplined procedure
    - Copy-and-Paste Paradigm
- Lack between Design Model and Implementation Model
- Design concepts get lost in the underlying model
- Short lifecycle of a Web Application -> Maintenance and Evolution issues -> Reuse issues

Web Systems: Problems

- Problems
    - Inability to maintain
    - Unable to meet evolving needs and grow at the rate needed – scaleability
    - Unreliable – crashes
- Web-dependent organizations cannot afford to have
    - Faulty systems – reliability, security issues
    - Frequent downtime – dependability
    - Wrong, inconsistent, or stale content/information
- Web systems problems are not easy to hide

Web Development Issues

- Many developers think that Web application development is just accomplished using "off the shelf" tools
- They have been taught to think this way
- Certain classes of applications do fit this simple generalization
- Many other Web applications do not
- "There is more to Web application development than visual design and user interface"
- Planning, system design, testing, continual maintenance, quality assurance, performance

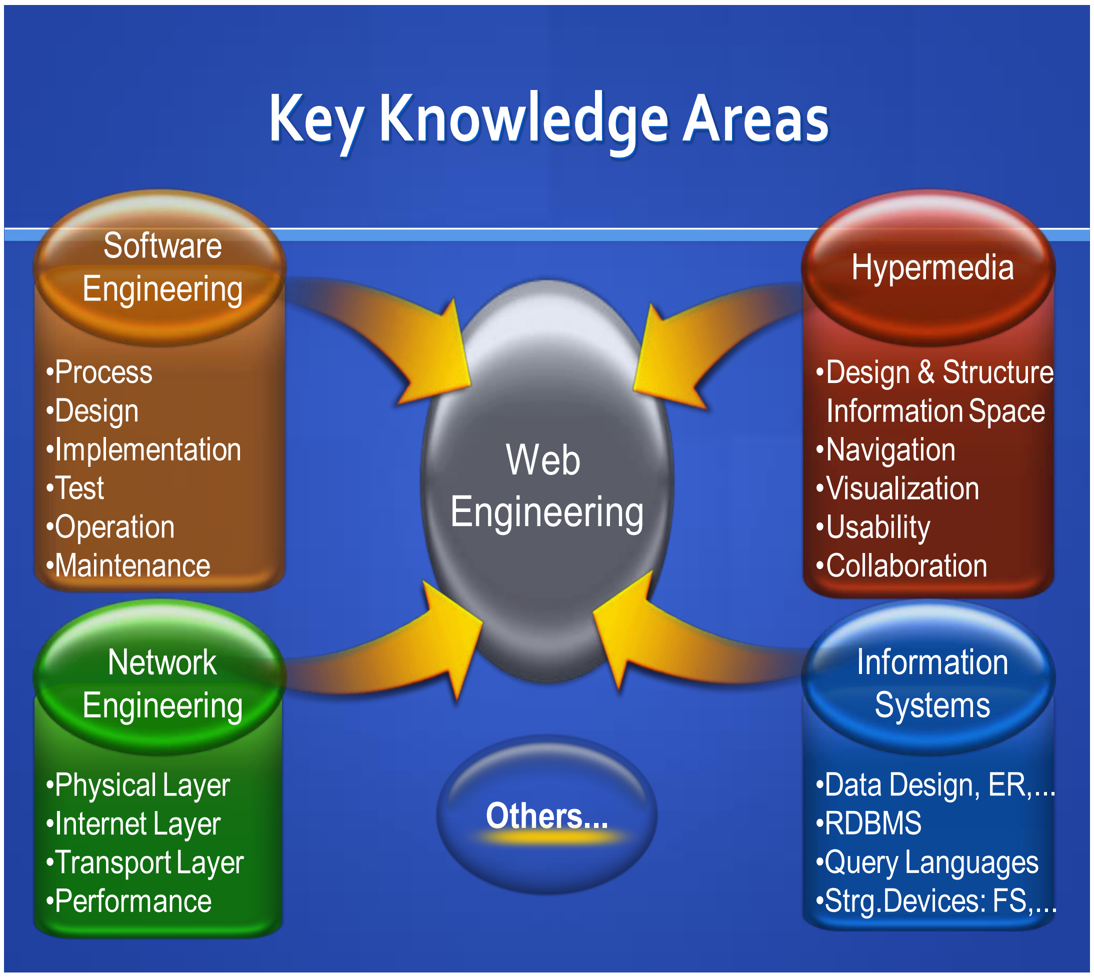
evaluation, scalability.

S.NO	RGPV QUESTION	YEAR	MARKS
Q.1	Write short notes on Forward Engineering.	Jun.2014,2013	5
Q.2	Write short notes on web Engineering.	Jun.2014,2013	5

UNIT-05/LECTURE-07

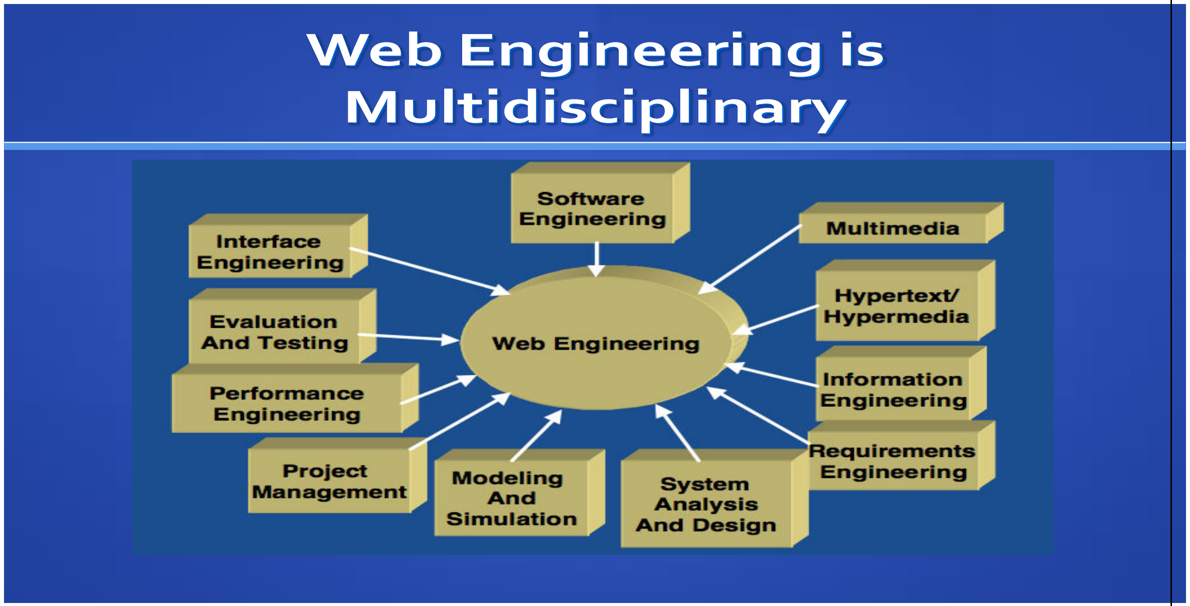
Goals of Web Engineering[RGPV/June 2013(7)]

- Develop high quality Web applications
    - Effective
    - Efficient
    - Achieve desired application
- Maintain and evolve
    - Plan for change – solution may change the problem
- Encourage the use of systematic, disciplined and quantifiable approaches and process models



Web Engineering Activities

- Requirements specification and analysis
- Web-based system analysis and design
- Web development methodologies and techniques
- Migration of legacy systems to Web environments
- Web-based real-time applications development
- Web-based multimedia application development
- Testing, verification and validation techniques and tools
- Quality assessment, control and assurance
- Management of access to applications and privileges
- Configuration and project management
- "Web metrics" – metrics for estimation of development effort
- Performance specification and evaluation
- Update and maintenance
- Development models, teams, and staffing
- Human and cultural aspects
- User-centric development
- Graphics, animation, and streaming
- Copyright, legal and social aspects



UNIT-05/LECTURE-08

Management

- The activities and tasks undertaken by one or more persons for the purpose of planning and controlling the activities of other in order to achieve objectives that could not be achieved by the others acting alone

Project Management[RGPV/June 2014(7)]

- A system of management procedures, practices, technologies, skills, and experience necessary to successfully manage an engineering project

Software Engineering Project Management

- Project management where the product is software

Universality of Management

- Management performs the same functions regardless of organizational position or enterprise
- Management functions are characteristic duties of all managers
- Management practices, methods, activities and tasks are specific to the enterprise

Issues with Software Engineering

- 70% of software organization have no defined methods
- Process are defined during the development
- Software ends up
  - Late
  - Over budget
  - Fails to meet requirements

Today's major problems with software development are not technical problems, but management problems.

Functions of management

- Planning
- Organizing
- Staffing
- Directing (leading)
- Controlling

Planning Activities[RGPV/June 2014(5)]

- Set objectives and goals
- Develop strategies
- Develop policies
- Forecast future situations
- Conduct a risk assessment
- Determine possible courses of action
- Make planning decisions
- Set procedures and rules
- Develop project plans
- Prepare budgets
- Document project plans

Organizing Activities

- Identify and group project function, activities, and tasks
- Select organizational structures
- Create organizational positions
- Define responsibilities and authority
- Establish position qualifications
- Document organizational decisions

Organizational Structure

- Conventional organization structure
  - Line organization
  - Staff organization
- Project organization structure
  - Functional

  - Project
  - Matrix
- Team Structure
  - Egoless
  - Chief programmer
  - Hierarchical

Issues In Staffing[RGPV/June 2014(5)]

- Lack of project management training
- Greatly varying skills
- Inability to predict productivity of engineers
- Lack of experience
- Turnover
- Not enough software engineers
  - Most graduates are theoretical
  - Or just coders

Staffing Activities

- Fill organizational positions
- Assimilate newly assigned personnel
- Educate or train personnel
- Provide for general development
- Evaluate and appraise personnel
- Compensate
- Terminate assignments
- Document staffing decisions

Directing Activities

- Provide leadership
- Supervise personnel
- Delegate authority
- Motivate personnel

- Build teams
- Coordinate activities
- Facilitate communication
- Resolve conflicts
- Manage changes
- Document directing decisions

Controlling Activities

- Develop standards of performance
- Establish monitoring and reporting systems
- Measure and analyze results
- Initiate corrective actions
- Reward and discipline
- Document controlling methods

S.NO	RGPV QUESTION	YEAR	MARKS
Q.1	Write short notes on Project Management Standards.	Jun.2014,2013	5
Q.2	Write short notes on web Engineering.	Jun.2014,2013	5

UNIT-04/LECTURE-09

Software Quality Assurance[RGPV/June 2013,2011(7)]

- Conformance to software requirements is the foundation from which software quality is measured.
- Specified standards are used to define the development criteria that are used to guide the manner in which software is engineered.
- Software must conform to implicit requirements (ease of use, maintainability, reliability, etc.) as well as its explicit requirements.

Quality

- Quality, simplistically, means that a product should meet its specification
- This is problematical for software systems
  - Tension between customer quality requirements (efficiency, reliability, etc.) and developer quality requirements (maintainability, reusability, etc.)
  - Some quality requirements are difficult to specify in an unambiguous way
  - Software specifications are usually incomplete and often inconsistent

Quality compromise

- We cannot wait for specifications to improve before paying attention to quality management
- Must put procedures into place to improve quality in spite of imperfect specification
- Quality management is therefore not just concerned with reducing defects but also with other product qualities

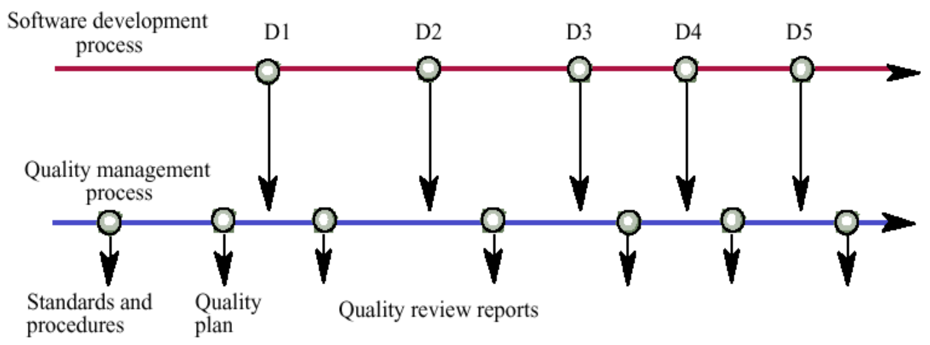
Quality management activities

- Quality assurance
  - Establish organisational procedures and standards for quality
- Quality planning
  - Select applicable procedures and standards for a particular project and modify these as required
- Quality control

- Ensure that procedures and standards are followed by the software development team

– Quality management should be separate from project management to ensure independence

Quality management and software development



©Ian Sommerville 2000 Software Engineering, 6th edition. Chapter 24 Slide 9

Quality assurance and standards

– Standards are the key to effective quality management

– They may be international, national, organizational or project standards

– Product standards define characteristics that all components should exhibit e.g. a common programming style

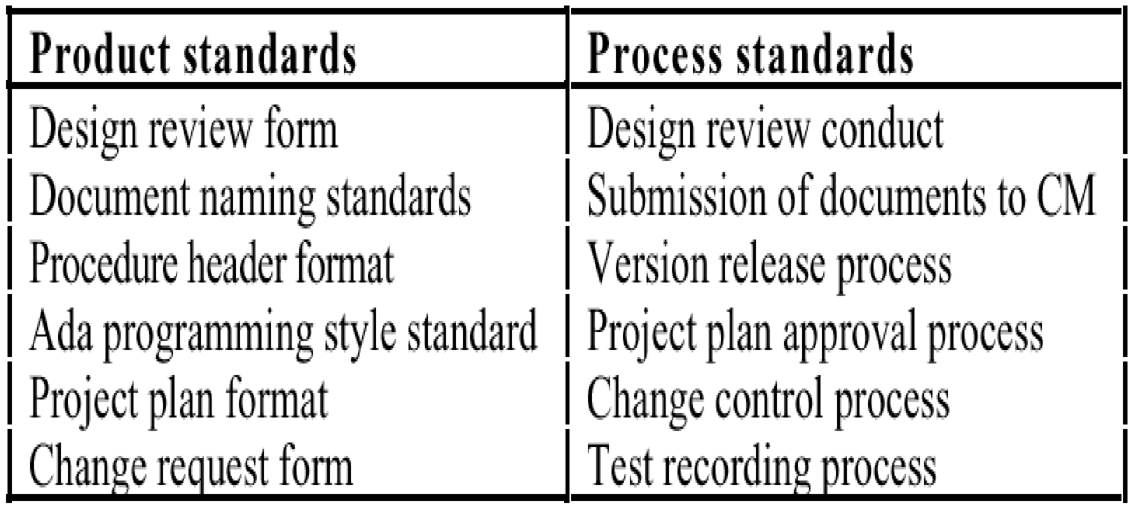
– Process standards define how the software process should be enacted

Importance of standards

– Encapsulation of best practice- avoids repetition of past mistakes

– Framework for quality assurance process - it involves checking standard compliance

– Provide continuity - new staff can understand the organisation by understand the standards applied.

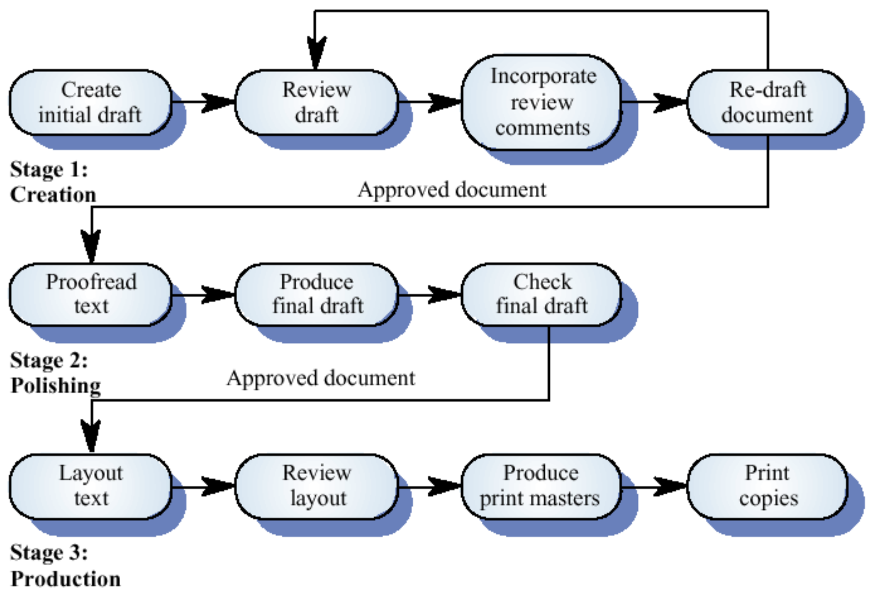


Product standards	Process standards
Design review form	Design review conduct
Document naming standards	Submission of documents to CM
Procedure header format	Version release process
Ada programming style standard	Project plan approval process
Project plan format	Change control process
Change request form	Test recording process

Problems with standards

- Not seen as relevant and up-to-date by software engineers
- Involve too much bureaucratic form filling
- Unsupported by software tools so tedious manual work is involved to maintain standards

Documentation process



©Ian Sommerville 2000 Software Engineering, 6th edition. Chapter 24 Slide 20

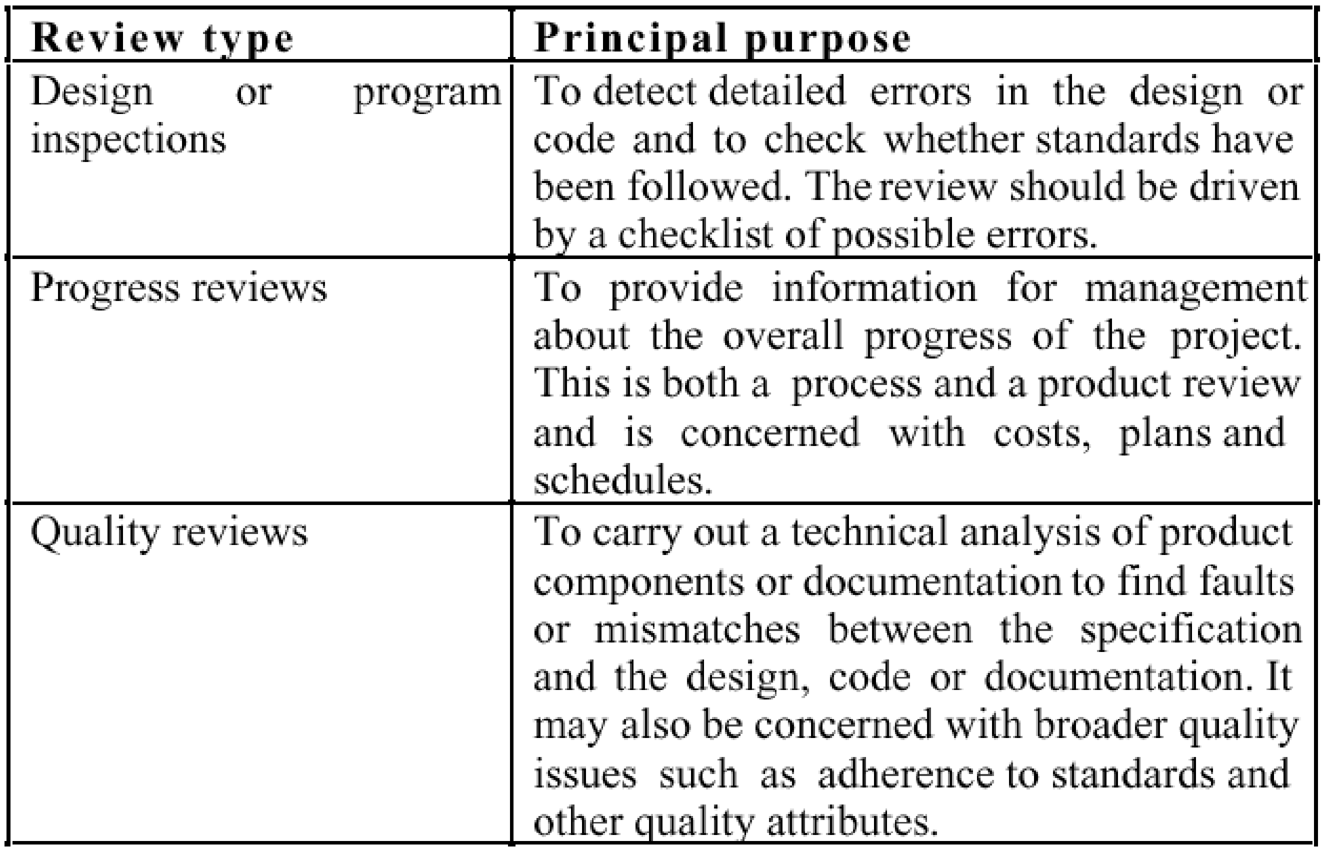
Quality control

- Checking the software development process to ensure that procedures and standards are being followed
- Two approaches to quality control
  - Quality reviews
  - Automated software assessment and software measurement

Quality reviews

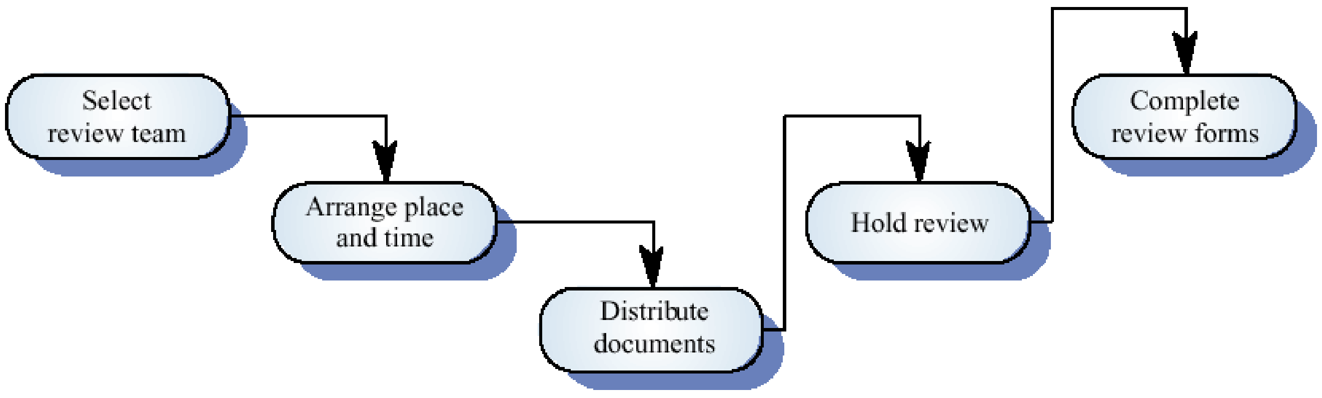
- The principal method of validating the quality of a process or of a product
- Group examined part or all of a process or system and its documentation to find potential problems
- There are different types of review with different objectives
  - Inspections for defect removal (product)
  - Reviews for progress assessment(product and process)
  - Quality reviews (product and standards)

Types of review



Review type	Principal purpose
Design or program inspections	To detect detailed errors in the design or code and to check whether standards have been followed. The review should be driven by a checklist of possible errors.
Progress reviews	To provide information for management about the overall progress of the project. This is both a process and a product review and is concerned with costs, plans and schedules.
Quality reviews	To carry out a technical analysis of product components or documentation to find faults or mismatches between the specification and the design, code or documentation. It may also be concerned with broader quality issues such as adherence to standards and other quality attributes.

Review Process



S.NO	RGPV QUESTION	YEAR	MARKS
Q.1	Explain Software Quality Assurance. Why should the quality assurance organization be independent of development organization?	Jun.2011	7
Q.2	Write short notes on Software Quality Assurance (SQA).	Jun.2011	5

REFERENCCE

BOOK	AUTHOR	PRIORITY
Software Engineering	P.S. Pressman	1
Software Engineering	Pankaj Jhalote	2