Software Engineering Tools

Syllabus

Unit	Contents	Lectures
3.	• **Software Engineering Tools** • Modeling • Analysis • Requirement Engineering • Work Breakdown Structure • Work Breakdown Structure Scheduling • Prototyping • CASE • I-CASE	5

Software Engineering Tools

- The software engineering tools that we are going to discuss are: -
  - Modelling
  - Analysis
  - Requirement Engineering
  - Work Breakdown Structure
  - WBS Scheduling
  - Prototyping
  - CASE

Modelling

- Abstract representation of a scenario or situation to understand and analyze.
- As it is an abstractions, considers those aspects and factors of the situation that are prominent, dominant and critical. Disregarding the above factors i.e. oversimplification of the complexity of the scenario, the model is not going to be a true representation.
- The model has to be the simplified representation close to the reality to be useful for the understanding and analysis.
- Model can represent static or dynamic scenario or situation.

Modelling …

- Static model shows structural details.
- Dynamic model shows the relationships and the nature of the interaction between different components of the model.
- Modeling is used as an Analysis Tool.
- Modeling is used as a System Representation Tool.
- Modeling helps to express complex ideas, improves the learning process and helps manipulation and communication between development teams.

Modelling …

- Software Engineering uses different models in the process of development. They are
  - Enterprise Model
  - Business Model
  - Data Model
  - Process Model

Analysis

- Process of deciphering the scenario to understand the
  - “What”
  - “Why”
  - “How”

of the different aspects of the scenario.

Analysis …

- Reveals
    - Relationship
    - Interactions
    - Behaviour
    - Resultant Outputs
- Analysis is the “Prerequisite” to build the “True Representation”.

Analysis …

- Good Analysis helps to
  - Understand Situation(s)
  - Brings Clarity in Cause and Effect Relationship
  - Reduce Ambiguity.
  - Prioritize various Critical Aspects to be built in the Model and Prototype.
- Analysis provides multiple views of the situation, providing better insight into the scenario or the situation.

Requirement Engineering

- Process of Ascertaining the Customer Requirements as described in terms of
  - Environment(s)
  - Goal(s)
  - Problem(s)
  - Expectation(s)
  - Solution(s)

Requirement Engineering …

- Puts the above ascertained and confirmed details into a well defined and structured document known as Requirement Definition and Description i.e. “RDD”.
- After the successful finalization of RDD, it is used to determine the Software Requirement Specification i.e. SRS.
- SRS provide(s) detail(s) of the Software Solution i.e. Software Architecture, Software Capability and the Required Platform to run the Solution (Software).

Requirement Engineering …

- Specifies the scope of the Software Solution along with the constraints and the conditions under which the software will operate.
- The Core Process(es) of Requirement Engineering is / are
  - Need Analysis
  - Problem Definition
  - Goal Determination
  - Solution Expectations
  - Requirement Definition and Description

Requirement Engineering …

- Entails the Software Engineer to Determine the Software Requirement Specification to solve the problem of the customer.
- Due to the involvement of multiple parties with different interest and focus, Requirement Engineering is a complex process.
- The parties that are involved are
  - End User(s) who actually operate the Software System
  - Customer whose business is affected by the software.
  - Stakeholders could be from within the organization or external to the organization.

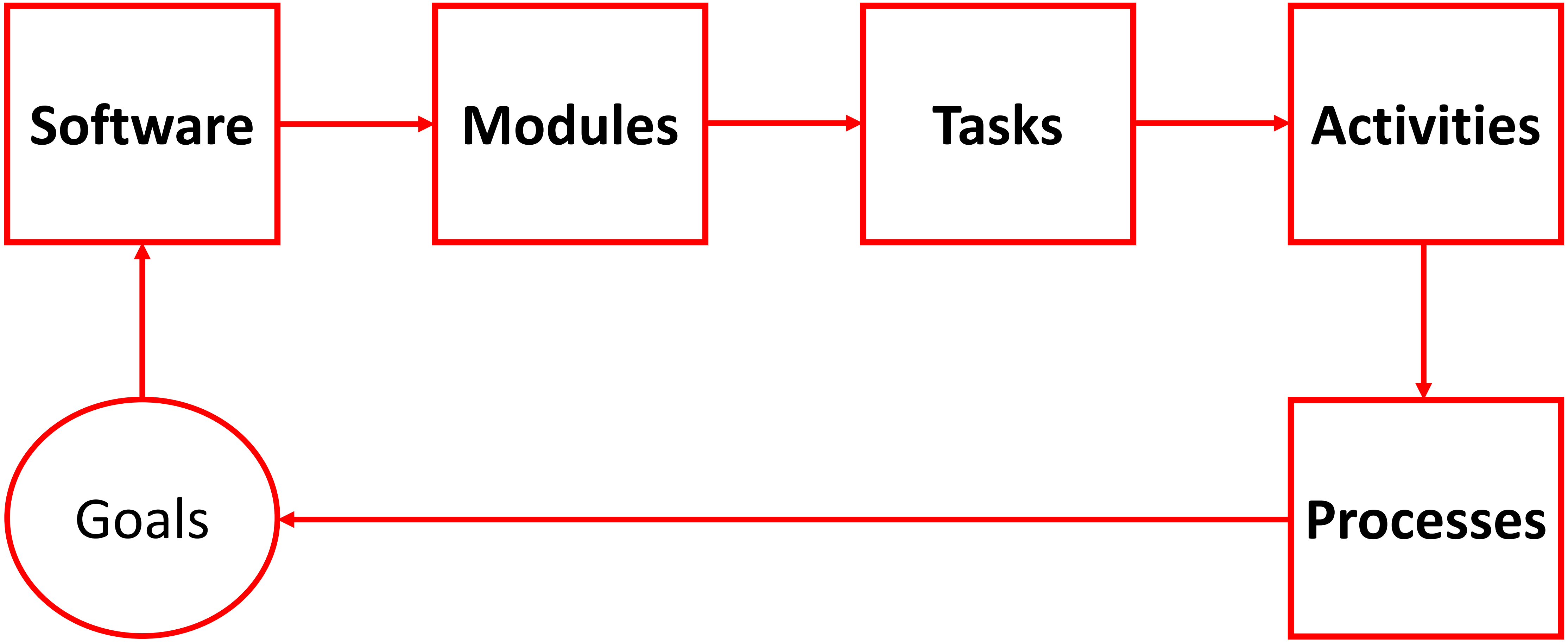
Requirement Engineering …

- The Requirement Engineering brings out **R**equirement **D**efinition and **D**escription i.e. RDD and **S**oftware **R**equirement **S**pecification i.e. SRS.
- SRS balance the multiple needs and interest of different participants (people).
- Both RDD and SRS is crucial for the successful completion of the Software Development with Quality Assurance to the Customer.

Work Breakdown Structuring

- Process of Breaking the Goal Driven Task into Components in such a way that when they are executed together, accomplish a Measurable Goal of the Task.
- The “Work” here is a Larger or Bigger Entity Broken Down into Components or Modules.
- Module(s) in Turn are Broken into Tasks.
- Task(s) in Turn are Broken into Activities.
- Activities in Turn are Broken into Processes;
- The WBS structure is given below: -

Work Breakdown Structuring Process(Structure)



Work Breakdown Structuring …

- Expert judgement and group consensus are tow down techniques.
- The work breakdown structure is bottom up approach.
- A work breakdown structure is a hierarchical char that accounts for an individual part of the system.
- A work breakdown structure chart indicates product hierarchy and process hierarchy.

Attributes of Good Work Breakdown Structuring

- Covers the entire scope and ensures that when it s completed, the desired goal of the software is achieved.
- Helps to plan and control the software development process effectively.
- Each unit of WBS has strong relationship with SRS and RDD.

How Work Breakdown Structuring is Done?

- No Method of constructing the WBS.
- It is advised to approach from Top to Bottom in a Systematic Manner and then Align it to the needs of the development strategy proposed for the development.
- WBS becomes the basis for
  - Scheduling
  - Monitoring
  - Tracking the Software Development Progress / Process

WBS Scheduling

- Once the WBS is finally built, it has to be scheduled to ensure that WBS units “Start on Time” and “Finish on Time” and complete the overall software delivery as per promise to the customer.
- Once scheduled, WBS Unit(s) act(s) as Tool to Monitor Progress and Control Result(s).
- Test Review(s) and Corrective Measure(s) are the actions that are / can be introduced at each level.

WBS Scheduling …

- WBS Scheduling in not as simple as just a simple exercise of putting WBS units in sequence against the time line.
- WBS Scheduling is governed by
  - The Logical Sequencing
  - Resource Availability
  - Delivery Requirement of The Customer
- WBS Scheduling can be rigid / flexible depending upon the interdependence on Parallel (other) Software Project(s).

GANTT Chart

- Devised in the mid 1895 by “Karol Adamiecki”, a Polish engineer with the desire to rope in the ideas and techniques of management to run the steelwork.
- Henry Gantt, an American engineer and project management consultant and Social Scientist, devised his own version of the chart in 1917 and hence got the name Gantt Chart.

GANTT Chart

- A Gantt chart is a horizontal bar chart developed as a production control tool in 1917 by Henry L. Gantt, an American engineer and social scientist.
- Frequently used in project management, a Gantt chart provides a graphical illustration of a schedule that helps to plan, coordinate, and track specific tasks in a project.

GANTT Chart

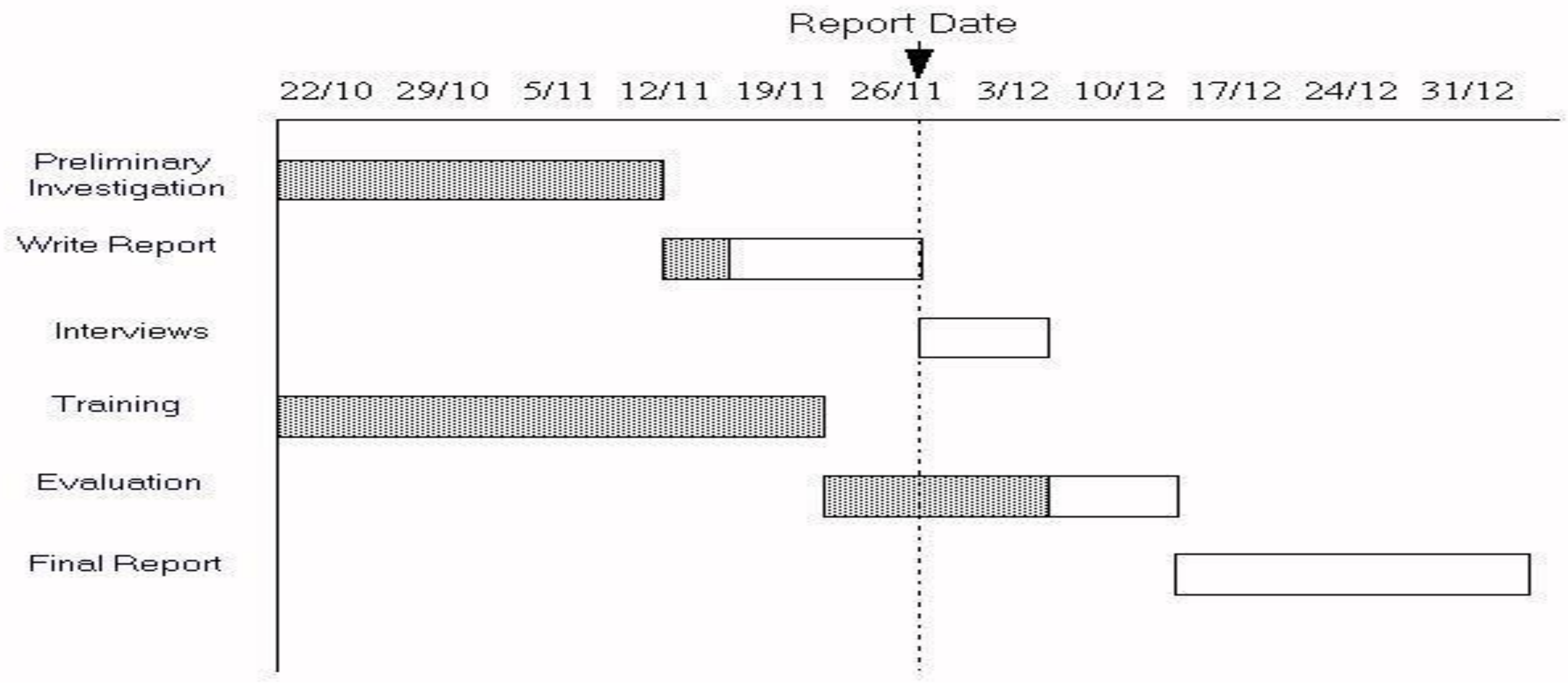


Figure 1: Gantt Chart

GANTT Chart ...

- Gantt charts are commonly used for tracking project schedules and for this it is required to be able to show additional information about the various tasks or phases of the project, like
  - How the tasks relate to each other?
  - How far each task has progressed?
  - What resources are being used for each task?

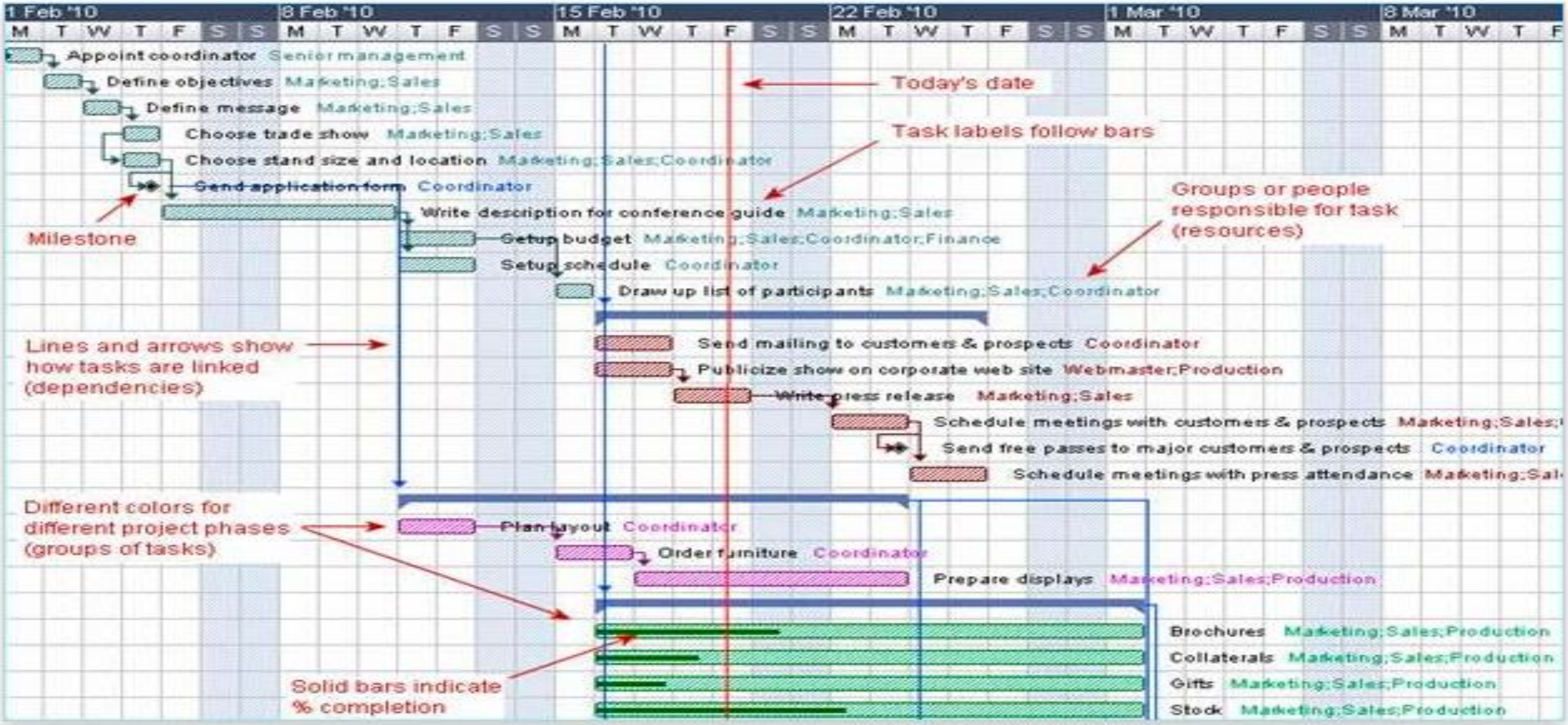
GANTT Chart …

- Used in project management.
- One of the most popular and useful ways of showing activities (tasks or events) displayed against time.
- On the left of the chart is a list of the activities and along the top is a suitable time scale.
- Each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity.

GANTT Chart …

- Gantt Chart allows us to find
  - What the various activities are
  - When each activity begins and ends
  - How long each activity is scheduled to last
  - Where activities overlap with other activities, and by how much
  - The start and end date of the whole project

GANTT Chart ...


1 Feb '10
8 Feb '10
15 Feb '10
22 Feb '10
1 Mar '10
8 Mar '10
M T W T F S S
Appoint coordinator Senior management
Define objectives Marketing;Sales
Define message Marketing;Sales
Choose trade show Marketing;Sales
Choose stand size and location Marketing;Sales;Coordinator
Send application form Coordinator
Write description for conference guide Marketing;Sales
Setup budget Marketing;Sales;Coordinator;Finance
Setup schedule Coordinator
Draw up list of participants Marketing;Sales;Coordinator
Send mailing to customers & prospects Coordinator
Publicize show on corporate web site Webmaster;Production
Write press release Marketing;Sales
Schedule meetings with customers & prospects Marketing;Sales;
Send free passes to major customers & prospects Coordinator
Schedule meetings with press attendance Marketing;Sal
Plan layout Coordinator
Order furniture Coordinator
Prepare displays Marketing;Sales;Production
Brochures Marketing;Sales;Production
Collaterals Marketing;Sales;Production
Gifts Marketing;Sales;Production
Stock Marketing;Sales;Production
Today's date
Task labels follow bars
Groups or people responsible for task (resources)
Milestone
Lines and arrows show how tasks are linked (dependencies)
Different colors for different project phases (groups of tasks)
Solid bars indicate % completion

Prototype

- Representative Model of an Entity.
- The Prototype could be of
    - Software
    - Process
    - Product
    - Technology
    - Tool(s)

made essentially to understand the requirement.

Prototype …

- Usually the customer and the End User(s) are not in a position to describe and specify their needs in clear and unambiguous terms , making software development a complex and difficult proposition.
- Prototype is developed and used to confirm whether the requirements as stated are right and understood correctly by the developer(s) or not.
- If the developer and customer are not sure of the approach, then a prototype can be constructed to demonstrate and test the choice of a solution.

Prototype …

- The use of prototype as a “Tool” helps to increase the validity of SRS and the Solution.
- The examples of Prototype are
  - The Pilot Project
  - Pilot Design
  - The Proof of the Concept or Experimentation

CASE i.e. Computer Aided Software Engineering

- Computer – Aided Software Engineering i.e. CASE is the domain of software tools used to design and implement applications.
- CASE tools are similar to and were partly inspired by computer-aided design (CAD) tools used for designing hardware products.
- CASE tools are used for developing High – Quality, Defect – Free, and maintainable software.
- CASE software is often associated with methods for the development of information systems together with automated tools that can be used in the software development process.

CASE i.e. Computer Aided Software Engineering

- Computer – Aided Software Engineering i.e. CASE is the tool to automate the certain process(es) in the Software Development Life Cycle.
- As automation is the methodology, it not only expedites the software development process but also provides a comprehensive approach to all aspects of software development.

CASE i.e. Computer Aided Software Engineering …

- Computer – Aided Software Engineering Software i.e. CASE Software is of three types
  - Tools
  - Workbench
  - Environment

CASE Tools

- CASE tools support specific tasks in the Software Development Life Cycle. Depending upon the attributes of usage and role in software development CASE Tools can be divided into the following categories:
    - Information Behaviour
    - Modelling
    - Planning and Scheduling
    - Analyse
    - Testing
    - Reverse – Engineering

CASE Tools …

- CASE tools and their usages

Tool	Usage
• Information Engineering	• Models business information in terms of primary objects, relationships and their application.
• Modeling	• Represents process, functions and data for understanding and analysis.
• Planning and Scheduling	• Creates WBS and its network. Estimate efforts, cost, resources and then schedules task for completion.
• Analyse	• Analyses risk, function and features.
• Testing	• Provides test management including creation of test case design, test cases and automating execution of testing process.
• Reverse – Engineering	• Reverse Engineering to find Structure, Design and Design Information.

CASE Tools …

- Battery of tools put together is called CASE Tools.
- When the tools are integrated as a set to provide an automation, the CASE Tools are termed as Integrated CASE i.e. I – Case Tools.
- I – case Tools are built by choice of individual tools to ensure that the information representation is consistent throughout the software development.

CASE Tools …

- Good I – CASE Tool provides assistance in
  - Analysis
  - Modelling
  - Prototyping
  - Specification Checking
  - High – End and Low – End Designing
  - Error – Detection
  - Testing
  - Documentation

CASE Tools …

- Good I – CASE Tool cuts down the efforts of the software development and reduces the impact of risk.
- Good I – CASE Tool forces you to dedicate more time in analysis and design.

Workbenches …

- Workbenches integrate two or more CASE tools and support specific software – process activities.
- Workbenches achieve: -
  - Homogeneous and Consistent Interface i.e. Presentation Integration.
  - Seamless Integration of Tools and Tool Chains i.e. Control and Data Integration.
- An example workbench is Microsoft's Visual Basic programming environment.

Workbenches …

- An example workbench is Microsoft's Visual Basic programming environment.
  - It incorporates several development tools:
    - GUI Builder
    - Editor
    - Debugger, etc.

Workbenches …

- We can classify Workbenches in the same manner as we have Tools;
  - Analysis Workbenches
  - Development Workbenches
  - Verification Workbenches, etc.

Environment

- An environment is a collection of CASE tools or workbenches that attempts to support the complete software process.
- Contrasts with tools that focus on one specific task or a specific part of the life-cycle.
- CASE Environments are classified as follows: -
  - Toolkits
  - Fourth generation Language
  - Language-centered
  - Integrated
  - Process-centered

Types of CASE Environment

- Toolkits
- Fourth Generation Language i.e. IDE (Integrated Development Environment)
- Language – Centered
- Integrated
- Process – Centered

Environment: Toolkit

- Loosely coupled collections of tools. These typically build on operating system workbenches such as the Unix Programmer's Workbench or the VMS VAX set. They typically perform integration via piping or some other basic mechanism to share data and pass control. The strength of easy integration is also one of the drawbacks. Simple passing of parameters via technologies such as shell scripting can't provide the kind of sophisticated integration that a common repository database can.

Environment: Fourth Generation

- These environments are also known as 4GL standing for fourth generation language environments due to the fact that the early environments were designed around specific languages such as Visual Basic. They were the first environments to provide deep integration of multiple tools. Typically these environments were focused on specific types of applications. For example, user-interface driven applications that did standard atomic transactions to a relational database. Examples are Informix 4GL, and Focus.

Environment: Language Centered

- Environments based on a single often object-oriented language such as the Symbolics Lisp Genera environment or VisualWorks Smalltalk from Parcplace. In these environments all the operating system resources were objects in the object-oriented language. This provides powerful debugging and graphical opportunities but the code developed is mostly limited to the specific language. For this reason, these environments were mostly a niche within CASE. Their use was mostly for prototyping and R&D projects. A common core idea for these environments was the model–view–controller user interface that facilitated keeping multiple presentations of the same design consistent with the underlying model. The MVC architecture was adopted by the other types of CASE environments as well as many of the applications that were built

Environment: Integrated

- These environments are an example of what most IT people tend to think of first when they think of CASE. Environments such as IBM's AD/Cycle, Andersen Consulting's FOUNDATION, the ICL CADES system, and DEC Cohesion. These environments attempt to cover the complete life-cycle from analysis to maintenance and provide an integrated database repository for storing all artifacts of the software process. The integrated software repository was the defining feature for these kinds of tools. They provided multiple different design models as well as support for code in heterogenous languages. One of the main goals for these types of environments was "round trip engineering": being able to make changes at the design level and have those automatically be reflected in the code and vice versa. These environments were

Environment: Process Centered

- This is the most ambitious type of integration. These environments attempt to not just formally specify the analysis and design objects of the software process but the actual process itself and to use that formal process to control and guide software projects. Examples are East, Enterprise II, Process Wise, Process Weaver, and Arcadia. These environments were by definition tied to some methodology since the software process itself is part of the environment and can control many aspects of tool invocation.

Analysis Tools

- Analysis tools are based on common premises that make them useful.
- The premises can be used to test the choice of tools.
- The basic premises are
  - Represent a situation or problem through complete information
  - Do not miss – out functions and features that are critical to the problem.
  - Point out completely conditions and constraints, internal as well as external affecting the problem.

Analysis Tools …

- Provide at least essential information on functions and behavior to help in design and implementation.
- Provide adequate information to structure the situation or problem in hierarchy (horizontal and vertical with clear designation on structural relations).
- Analysis tools help in two main areas of the software development
    - Requirements Engineering
    - Modeling

Analysis Tools …

Requirement Engineering	• Understanding • Scoping • Description • Definition • Specification • Prioritisation
Modeling	Modeling the requirement in terms of • Data • Function • Behavior for better design and architecture; first through prototype and then in detail design

Information Analysis

- A situation or problem is made of multiple data types
  - Numbers
  - Characters
  - Images
  - Voice
  - Events
- The event shows the status such as on or off, complete or incomplete, success or failure.

Information Analysis …

- These put together in context provides information of interest to the software developer.
- The micro analysis of the above details helps to understand data, data definitions and structure.
- It helps to understand at higher level the information relationships, flow and its structure.
- Data and events, put together in context, form information.
- Events are generated through data manipulation i.e. acquisition, validation, computation and communication through flow.

Information Analysis …

- In an order processing system,
  - Bill is an object
  - Customer Name, Product Name, Quality, Price, Date of Dispatch, Terms of Payment, Payment Amount are the data content in the object.
  - All these data flows in the system from different sources i.e. product catalogue, customer order etc.
  - Billing the dispatch to the customer is an event.
  - Delivery of goods indicated by delivery date is a control to trigger the event, billing process.

Work Breakdown Structure (WBS)

- WBS is used to break the RDD into smaller units or components for planning and budgeting and for controlling cost, resources and efforts.
- WBS has to detailed enough to monitor and track the progress of the development.
- Good WBS is the one that is evolved after considering the development strategy and an implementation strategy and s synchronized with steps in the software development cycle.
- WBS maps RDD with SRS.

Work Breakdown Structure (WBS)

- WBS has to cover complete SRS for the execution.
- WBS has to be SRS oriented and not task oriented.
- WBS is a hierarchy of elements that decomposes the software development plan based on RDD into discrete work tasks. It should provide information on
  - Inclusion of significant, critical work elements traceable to SRS and RDD
  - Repository to achieve results
  - Framework for scheduling, budgeting, monitoring and tracking control of cost, efforts and schedule.

Parameter of Work Breakdown Structure (WBS)

- The parameters are
  - System structure using SSAD and OOSAD
  - Functions
  - Organisational Units
  - Life Cycle Phases
- The choice of WBS parameters depends upon the nature, complexity and the development and implementation strategy.

WBS with System Structure as Parameter

- If the system structure is broken into modules then they become the anchor to build WBS.
- Linked with delivery and implementation plan

WBS with Functions as Parameter

- If the system is largely function driven, then WBS can be built around its application, which is designed to deliver functions.

WBS with Organisational Units as Parameter

- If the system can be visualised after modeling by structure and / or function(s) into organisational units, then WBS can be anchored onto organisational units.

WBS with Life Cycle Phases as Parameter

- We can follow life cycle phase and build WBS for each phase starting form inception till delivery to the customer.

Guidelines to Choose Appropriate Parameters for WBS

Software System	Size	RDD & SRS	Complexity Driven	Function Driven	Technology Parameter	WBS
A	Small Medium	Stable	Simple	Yes	No	System Structure
B	Large	Unstable Needs Prototype	Complex	No	Yes	Life Cycle
C	Medium	Stable	Not so Complex	Yes	Yes	Life Cycle
D	Large	Stable	Complex	Yes	Yes	System Structure

Guidelines to Choose Appropriate Parameters for WBS

Software System	Size	RDD & SRS	Complexity Driven	Function Driven	Technology Parameter	WBS
E	Large	Not Stable but Controllable	Complex Multiple Locations Technology Specific	Yes	Yes	Organisational Unit

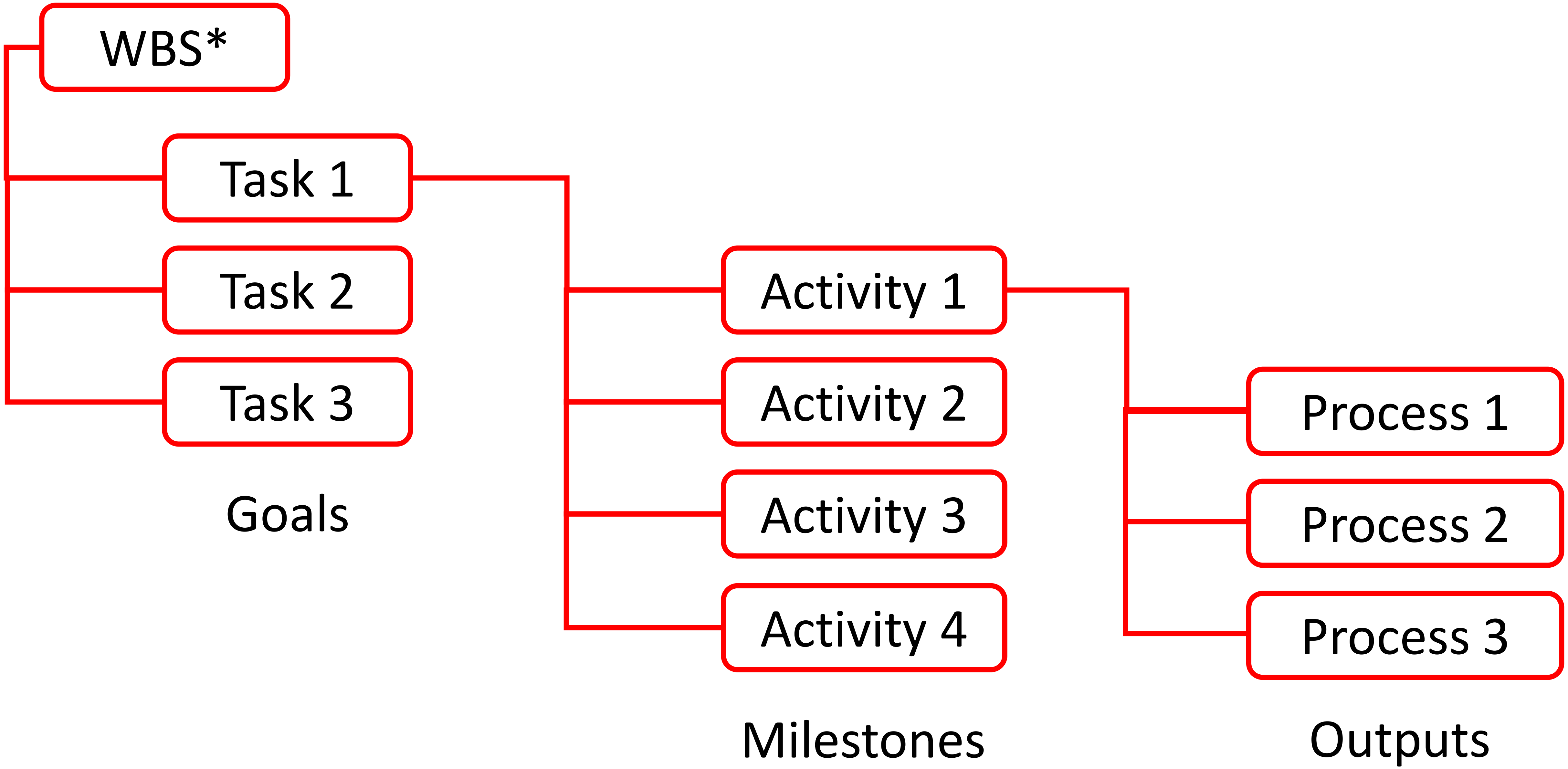
Characteristics of WBS

- Each WBS is well defined statement having specific goals to be achieved.
- The activities in WBS are standard Life Cycle activities, which when completed achieve the goals.
- Each WBS has milestones, suggesting a goal that can be traced back to SRS.
- Goals and Milestones should have to be quantifiable, measurable and controllable.

Characteristics of WBS …

- The breakdown of WBS into sub – activities has to be such that the scheduling is easier, resource allocation is possible and responsibility to complete the activity can be fixed with an individual.
- The activities are such that their performance can be measured to control planned effort, resources, cost and time allocation.

Characteristics of WBS …



WBS Model

- WBS is modeled / documented using GANTT Chart.
- The software tools that are used to model WBS are MS. Project and Prima Vera.
- Both MS. Project and Prima Vera provide support to plan, schedule and control software development irrespective to the choice of WBS Building Parameters.

Prototyping

- SE Tool used to provide feedback on tools and technologies to customer, end users and stakeholders.
- Used to improve communication requirements between developers and customer(s), end user(s) and stakeholder(s).
- There can be a situation where end user(s) find it difficult to visualize how a software solution will work for critical processes even when the RDD and SRS is documented through graphics. In such case, prototype becomes very handy to provide the customer with good feel of the software solution and it meet his / her requirements.

Benefits Associated with Prototyping

- Correlating requirements elicitation to requirement analysis
- Obtaining meaningful feedback on alternative software solution
- Confirming the technical feasibility of a required (key) function and feature.

Drawbacks Associated with Prototyping

- The customer(s), end user(s) and stakeholder(s) may lead to disillusioned by forming an impression that the prototype is too simple and may not lead to the delivery of the required / desired solution.
- Customer(s), end user(s) and stakeholder(s) may ask for comparatively more versatile and complex prototype that may lead to the increase in cost of the development.

How to Develop Prototype?

- Establish goals for the prototype
- Define scope of the prototype, its functionality and features
- Develop the prototype
- Execute the prototype
- Evaluate the prototype through the feedback from the end user(s)
- Restate the requirements incorporating changes in requirement, design, architecture

Methods and Tools Associated with Prototyping

- Use of Fourth Generation Techniques
- Use of Reusable Software
- Use of Formal Specification Language and Tools

Methods and Tools Associated with Prototyping …

- Use of Fourth Generation Techniques
  - Query Language
  - Report Writers and Generators
  - Program and Application Generator
  - High – End Object Oriented Programming Languages
- Use of 4GTs saves software specific coding time and allows rapid development.

Methods and Tools Associated with Prototyping …

- Use of Reusable Software
  - In order to save development time, the reusable software components are to be used and assembled to build the prototype.
  - This approach eliminates the coding required to build the prototype.
  - Reusable component library has to developed

Methods and Tools Associated with Prototyping …

- Use of Formal Specification Language and Tools
  - CSP, LARCH, VDM and Z are some of the Formal Specification Languages.
  - Formal Specification Language has following components
    - Syntax that defines specific notation to represent SRS
    - Semantics to define the object to describe the system
    - Set of Relations that defines the rules to indicate which object meet the SRS

CASE and I – CASE Tools

- CASE and I – CASE Tools are set of tools that allow the software developer to automate the manual activities and improve insight into RDD and SRS.
- The CASE approach or system has a variety of tools that provide assistance to automate the following tasks in the SDLC
  - Requirement Analysis
  - SSAD
  - OOSAD
  - Prototyping
  - Design Specification Checking

CASE and I – CASE Tools …

- Logical data Modelling
- Database and File Design
- Testing
- Programing
- Project Management

Characteristics of CASE and I – CASE Tools

- Graphic Interface to Design
  - Diagrams
  - Charts
  - Models
    - Uppercase
    - Middlecase
    - Lowercase

Characteristics of CASE and I – CASE Tools …

- Information Repository, a Data Dictionary for
  - Efficient Information Management
  - Selection
  - Usage
  - Application
  - Storage

Characteristics of CASE and I – CASE Tools …

- Common User Interface for Integration of Multiple Tools used in various phase
- Automatic Code Generators
- Automatic Testing Tools

CASE Tools by Application

- CASE tools can be classified by its use, function and perspective.
- The three criteria are: -
  - Use
  - Process
  - Integration
- CASE & I – CASE tools provide automated development support to the software development team(s).
- CASE & I – CASE Tools enforce application of standard, principle and best practices.

CASE Tools by Application …

- The intelligent application of CASE and I – CASE Tools is possible if and only if the software development team is creative and has domain knowledge in all the aspects of SE.
- The real challenge is to organize different tools in such a way that they produce effective support in terms of
  - Speedy Development
  - Quality Improvement
  - Ease in Change Management

CASE Tools by Application …

Application	CASE Tool	Purpose of The Tool
• Planning	• Excel Spreadsheet • MS. Project • PERT / CPM Network	• Functional Application • Planning • Scheduling • Control
• Editing	• Text Editors • Word Processors	• Speed & Efficiency
• Testing	• Test Data Generators	• Speed & Efficiency
• Prototyping	• UML	• Preparation of RDD & SRS
• Documentation	• Report Generator • Publishing • Imaging • PPT Presentation	• Faster Structural Documentation with Quality of Presentation

CASE Tools by Application …

Application	CASE Tool	Purpose of The Tool
• Programming & Language Processing	• Program Generator • Code Generator • Compiler • Interpreter	• Error Free Optimized Programs
• Integration	• Interface • Connectivity	• System Integration
• Template		• Guided Systematic Development