Software Engineering and Project Management

(Code : 314443)

Semester V - Information Technology

(Savitribai Phule Pune University)

Strictly as per the New Credit System Syllabus (2015 Course)
Savitribai Phule Pune University w.e.f. academic year 2017-2018

S. P. Kosbatwar

Ph.D. (Persuing)

Smt. Kashibai Navale College of Engineering,

Vadgaon Budruk, Off Sinhgad Road,
Pune - 411041

Sneha Manoj Patil

ME (CSE)

Smt. Kashibai Navale College of Engineering,

Vadgaon Budruk, Off Sinhgad Road,
Pune - 411041

V. S. Phad

ME (CSE)

Smt. Kashibai Navale College of Engineering,

Vadgaon Budruk, Off Sinhgad Road,
Pune - 411041

M. A. Ansari

M-Tech (CSE)

Smt. Kashibai Navale College of Engineering,

Vadgaon Budruk, Off Sinhgad Road,

Pune - 411041

Tech-Max Publications, Pune

Innovation Throughout Engineering Division

PO276A

Software Engineering and Project Management

S. P. Kosbatwar, Sneha Manoj Patil, V. S. Phad, M. A. Ansari

Semester V - Information Technology (Savitribai Phule Pune University)

First Printed in India : September 2006 **(Pune University)**
First Edition : June 2017

This edition is for sale in India, Bangladesh, Bhutan, Maldives, Nepal, Pakistan, Sri Lanka and designated countries in South-East Asia. Sale and purchase of this book outside of these countries is unauthorized by the publisher.

Printed at : Image Offset, Dugane Ind. Area, Survey No. 28/25, Dhayari, Near Pari Company, Pune - 41, Maharashtra State, India. E-mail : rahulshahimage@gmail.com

ISBN 978-93-5224-580-2

Published by

Tech-Max Publications

Head Office : B/5, First floor, Maniratna Complex, Taware Colony, Aranyeshwar Corner, Pune - 411 009. Maharashtra State, India.
Ph : 91-20-24225065, 91-20-24217965. Fax 020-24228978. Email : info@techmaxbooks.com,
Website : www.techmaxbooks.com

[314443] (FID : TP485) (Book Code : PO276A)

Preface

Dear students,

We are extremely happy to present book on **"Software Engineering and Project Management"** for you, the Information Technology. We have divided the subject into small chapters so that the topics can be arranged and understood properly. The topics within the chapters have been arranged in a proper sequence to ensure smooth flow of the subject.

We are thankful to Shri. Pradeep Lunawat and Shri. Sachin Shah for the encouragement and support that they have extended. We are also thankful to the staff members of Tech-Max Publications and others for their efforts to make this book as good as it is. We have jointly made every possible efforts to eliminate all the errors in this book. However if you find any, please let us know, because that will help us to improve further.

We are also thankful to my family members and friends for patience and encouragement.

- Authors

□□□

Syllabus

314443 : Software Engineering and Project Management

Teaching Scheme	Credits	Examination Scheme
Lectures : 03 Hours/Week	03	In-Semester : 30 Marks
		End-Semester : 70 Marks

Prerequisite

1. Problem solving and object oriented programming.
2. Fundamental of data structures.

Course Objectives

1. To understand the nature of software complexity in various application domains, disciplined way of software development and software lifecycle process models.
2. To introduce principles of agile software development, the SCRUM process and agile practices.
3. To know methods of capturing, specifying, visualizing and analyzing software requirements.
4. To understand project management through life cycle of the project.
5. To understand current and future trends and practices in the IT industry.
6. To learn about project planning, execution, tracking, audit and closure of project.

Course Outcomes

1. To identify unique features of various software application domains and classify software applications.
2. To choose and apply appropriate lifecycle model of software development.
3. To describe principles of agile development, discuss the SCRUM process and distinguish agile process model from other process models.
4. To analyze software requirements by applying various modeling techniques.
5. To list and classify CASE tools and discuss recent trends and research in software engineering.
6. To understand IT project management through life cycle of the project and future trends in IT Project Management.

Unit I - Introduction to Software Engineering (06 Hours)

Nature of Software, Software Process, Software Engineering Practice, Software Myths, Generic Process model, Analysis and comparison of Process Models: Waterfall Model, Incremental Models, Evolutionary Models, Concurrent, Specialized Process Models, Personal and Team Process Models, Introduction to Clean Room Software Engineering.

Software Quality Assurance (SQA) : Verification and Validation, SQA Plans, Software Quality Frameworks, ISO 9000 Models, CMM Models. **(Refer Chapters 1, 2 and 3)**

Unit II - Requirement Analysis (06 Hours)

Requirements Capturing : requirements engineering (elicitation, specification, validation, negotiation, prioritizing requirements (Kano diagram) - real life application case study.

Requirements Analysis : basics, scenario based modeling, UML models: use case diagram and class diagram, data modeling, data and control flow model, behavioral modeling using state diagrams - real life application case study, software Requirement Specification. **(Refer Chapters 4 and 5)**

Unit III - Project Planning (06 Hours)

Project initiation, Planning Scope Management, Creating the Work Breakdown Structure, Effort estimation and scheduling: Importance of Project Schedules, Estimating Activity Resources, Estimating Activity Durations, Developing the Schedule using Gantt Charts, Adding Milestones to Gantt Charts, Using Tracking Gantt Charts to Compare Planned and Actual Dates, Critical Path Method, Program Evaluation and Review Technique (PERT) with examples. Planning Cost Management, Estimating Costs, Types of Cost Estimates, Cost Estimation Tools and Techniques, Typical Problems with IT Cost Estimates. **(Refer Chapter 6)**

Unit IV - Agile Development Process (06 Hours)

Agile Development : Agile manifesto, agility and cost of change, agility principles, myth of planned development, toolset for the agile process.

Extreme Programming : XP values, process, industrial XP, SCRUM - process flow, scrum roles, scrum cycle description, product backlog, sprint planning meeting, sprint backlog, sprint execution, daily scrum meeting, maintaining sprint backlog and burn-down chart, sprint review and retrospective.

Agile Practices: test driven development, refactoring, pair programming, continuous integration, exploratory testing versus scripted testing **(Refer Chapters 7 and 8)**

Unit V - Project Management (06 Hours)

Project monitoring and control : tools for project management, Software tools like Microsoft project management or any other open source tools.

The Importance of Project Quality Management : Planning Quality Management, Performing Quality Assurance, Controlling Quality, Tools and Techniques for Quality Control (statistical control, six sigma)

The Importance of Project Risk Management, Planning Risk Management, Common Sources of Risk in IT Projects. **(Refer Chapter 9, 10 and 11)**

Unit VI - Recent Trends in Software Engineering and Project Management (06 Hours)

Software configuration management : SCM basics, SCM repository, SCM process, SCM tools such as GitHub, CASE – taxonomy, tool-kits, workbenches, environments, components of CASE, categories (upper, lower and integrated CASE tools).

Emerging software engineering trends : technology evolution, process trends, collaborative development, test-driven development, global software development challenges

Project Management trends : CRM, ERP: Basic concepts, Advantages and limitations, SAP, Business process reengineering, International Project Management, Case studies. **(Refer Chapters 12 and 13)**

❑❑❑

UNIT I

Syllabus : Nature of Software, Software Process, Software Engineering Practice, Software Myths, Generic Process model, Analysis and comparison of Process Models: Waterfall Model, Incremental Models, Evolutionary Models, Concurrent, Specialized Process Models, Personal and Team Process Models, Introduction to Clean Room Software Engineering. Software Quality Assurance (SQA) : Verification and Validation, SQA Plans, Software Quality Frameworks, ISO 9000 Models, CMM Models.

Chapter 1 : Introduction to Software Engineering 1-1 to 1-11

Chapter 2 : Process Models 2-1 to 2-23

UNIT II

Syllabus : Requirements Capturing : requirements engineering (elicitation, specification, validation, negotiation, prioritizing requirements (Kano diagram) - real life application case study. Requirements Analysis : basics, scenario based modeling, UML models: use case diagram and class diagram, data modeling, data and control flow model, behavioral modeling using state diagrams - real life application case study, software Requirement Specification.

UNIT III

Syllabus : Project initiation, Planning Scope Management, Creating the Work Breakdown Structure, Effort estimation and scheduling: Importance of Project Schedules, Estimating Activity Resources, Estimating Activity Durations, Developing the Schedule using Gantt Charts, Adding Milestones to Gantt Charts, Using Tracking Gantt Charts to Compare Planned and Actual Dates, Critical Path Method, Program Evaluation and Review Technique (PERT) with examples. Planning Cost Management, Estimating Costs, Types of Cost Estimates, Cost Estimation Tools and Techniques, Typical Problems with IT Cost Estimates.

UNIT IV

Syllabus : Agile Development: Agile manifesto, agility and cost of change, agility principles, myth of planned development, toolset for the agile process. Extreme Programming: XP values, process, industrial XP, SCRUM - process flow, scrum roles, scrum cycle description, product backlog, sprint planning meeting, sprint backlog, sprint execution, daily scrum meeting, maintaining sprint backlog and burn-down chart, sprint review and retrospective. Agile Practices: test driven development, refactoring, pair programming, continuous integration, exploratory testing versus scripted testing

Chapter 7 : Agile Development Process 7-1 to 7-11

Chapter 8 : Extreme Programming 8-1 to 8-16

UNIT V

Syllabus : Project monitoring and control: tools for project management, Software tools like Microsoft project management or any other open source tools. The Importance of Project Quality Management: Planning Quality Management, Performing Quality Assurance, Controlling Quality, Tools and Techniques for Quality Control (statistical control, six sigma) The Importance of Project Risk Management, Planning Risk Management, Common Sources of Risk in IT Projects.

Chapter 9 : Project Monitoring and Control 9-1 to 9-8

Chapter 10 : Project Quality Management 10-1 to 10-10

UNIT VI

Syllabus : Software configuration management: SCM basics, SCM repository, SCM process, SCM tools such as GitHub, CASE – taxonomy, tool-kits, workbenches, environments, components of CASE, categories (upper, lower and integrated CASE tools). Emerging software engineering trends: technology evolution, process trends, collaborative development, test-driven development, global software development challenges Project Management trends: CRM, ERP: Basic concepts, Advantages and limitations, SAP, Business process reengineering, International Project Management, Case studies.

❑❑❑

UNIT I

CHAPTER 1

Introduction to Software Engineering

Syllabus

Nature of Software, Software Process, Software Engineering Practice, Software Myths.

Syllabus Topic : Introduction to Software Engineering

1.1 Introduction to Software Engineering SPPU - May 12, Dec. 12

University Questions

Q. What is software Engineering ?

(May 2012, 3 Marks)

Q. Define software engineering.

(Dec. 2012, 4 Marks)

Review Questions

Q. Why Computer Software is important ?

Q. Explain the term Computer Software in brief.

- Computer software has become an integral part of our daily lives. It helps in all sorts of businesses and decision makings in business. The application of computer software includes : Telecommunication, transportation, military, medical sciences, online shopping, entertainment industry, office products, education industry, construction business, IT industry, banking sector and many more.
- The software applications have a great impact on our social life and cultural life as well. Due to its widespread use, it is the need of time to develop technologies to produce high quality, user friendly, efficient and economical software.
- Computer software is actually a product developed by software engineers by making use of various software engineering processes and activities.
- Software consists of data, programs and the related documents. All these elements build a configuration that is created as a part of the software engineering process. The main motive behind software engineering is to give a framework for building software with better quality.
- We can say that the software has become the key element in all computer based systems and products.

Syllabus Topic : Nature of Software

1.2 Nature of Software

Review Questions

Q. List and explain the characteristics that describe the nature of software.

Q. Describe the drawback of software.

- The nature of software has great impact on software engineering systems. The general nature of software may describe the important characteristic : **Reliability**. If we consider the use of software in air traffic

control system and space shuttle, then there should be not be any chances of failure of software. In these examples, if reliability is not taken into consideration, then the human life is at risk.

- In addition to this, there are four other important characteristics that describe the nature of software :

1. Absence of fundamental theory
2. Ease of change
3. Rapid evolution of technologies
4. Low manufacturing cost

1.2.1 Absence of Fundamental Theory

- If we consider the example of physics, we see there are fundamental laws of physics, but in software there are no such fundamental laws despite the researches done by the computer scientists.
- Because of this drawback, it is very difficult to do any reasoning about the software until it is developed. Thus the developers practice few software engineering standards that are not foolproof. But the codes written for developing software are following the discipline and some solid principles.

1.2.2 Ease of Change

- The software has the provision to be altered at any stage of time. Thus the software development organizations take the advantage of this feature. From the customer's point of view, the change is always required throughout its development cycle and even after its delivery to the customer.
- Since there are no basic rules of software, there is no rule available to accommodate the changes and its impact until it is developed. Thus we can say that the ease of change is a gift of God to the developers and the development organizations.

1.2.3 Rapid Evolution of Technologies

- In the modern age, the software development technologies and development environments are changing rapidly with an extreme speed. It becomes the need of the time to keep the software engineers updated and armed with latest technologies and skills.
- The software engineering standards must also be revised with the technology evolutions.

1.2.4 Low Manufacturing Cost

- The cost of software reproduction and installation is less as compared to a new development. Today nearly 80 percent of the software development contains only maintenance and only 20 percent is new development. This reflects that manufacturing a product is considerably involves very low cost.
- The software reusability is an important characteristic that benefits the development organizations a lot in manufacturing the software at low cost.

1.3 Software Engineering : A Layered Technology

SPPU - May 14

University Question

Q. Software Engineering is considered as layered technology. Comment. **(May 2014, 9 Marks)**

Review Questions

Q. Define the term Software Engineering.

Q. Define Software Engineering as Layered Technology.

Q. How Software Characteristics are different from Hardware Characteristics ?

Q. Define the terms Software Crisis.

Q. Define the terms Legacy Software.

Q. Define software engineering. What are the characteristics of software which make its Engineering different from Industrial Manufacturing ?

Q. What are the problems associated with development process ? (Hint : Refer Software Crisis).

- The term software engineering is defined as :

  "By using the principles of sound engineering and its establishment, software is developed that should be economical and should work efficiently on real machines."

- Software engineering is a systematic and disciplined approach towards developments of software. The approach must be systematic for operation of the software and the maintenance of it too.

- Software engineering is considered as a layered technology. These layers includes :

1. Quality focus	2. Process
3. Methods	4. Tools

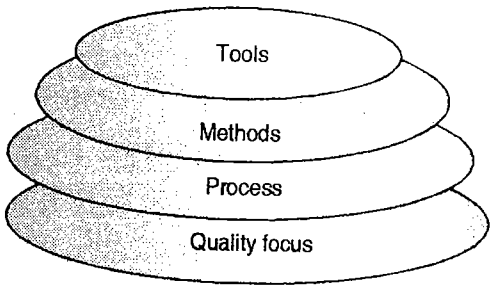


Fig. 1.3.1 : Software engineering layers

1.3.1 Quality Focus

Quality is nothing but **'degree of goodness'**. Software quality cannot be measured directly. Good quality software has following characteristics :

1. **Correctness** is degree to which software performs its required function.
2. **Maintainability** is an ease with which software is maintained. Maintenance of software implies change in software. If software is easy to maintain, then the quality of that software is supposed to be good.
3. **Integrity** is a security provided so that unauthorized user can not access data or information. e.g. password authentication.
4. **Usability** is the efforts required to use or operate the software.

1.3.2 Process

Software process defines a framework that includes different activities and tasks. In short, process defines following **'what'** activities and tasks for software development :

1. What activities are to be carried out ?
2. What actions will be taken ?
3. What tasks are to be carried out in a given action ?

1.3.3 Methods

Method provides technical way to implement the software i.e. **'how to'** implement. Methods consist of collection of tasks that include :

1. **Communication :** Between customer and developer.
2. **Requirement analysis** : To state requirements in detail.
3. **Analysis and design modelling** : To build a prototyping model of software to exhibit the requirements clearly.
4. **Program construction :** Implementation of requirements using conventional programming languages or automated tools.
5. **Testing and support :** Test for errors and expected results.

1.3.4 Tools

Software tool is an automated support for software development.

For example

1. Microsoft front page or Microsoft Publisher can be used as web designing tool.
2. Rational Rose can be used as object oriented analysis and design tool.

1.3.5 The Characteristics of Software

SPPU-May 12, Dec. 12, May 13

University Questions

Q. What are the characteristics of software ?
(May 2012, May 2013, 3 Marks)

Q. What are the software characteristics ?
(Dec. 2012, 3 Marks)

- Software is having some characteristics, those are totally different from hardware characteristics or the industrial manufacturing :
  - Software is developed or engineered and it is not manufactured like other hardware products.
- There exist some similarities between development of software and manufacturing of hardware :
  - In both the cases quality is achieved by good design but manufacturing phase of hardware can introduce quality problems that are absent in software.
  - Both activities depend on people but relationship between people applied and work done is different in both the cases.
  - Both the cases require the 'construction of product', but approaches are different.
  - Software does not wear out.
  - Note that, wear out means process of losing the material.
  - Hardware components are affected by dust, temperature and other environmental maladies. Stated simply, hardware can wear out. However software does not influenced by such environmental means.
  - When hardware component wear out, it can be replaced by spare part. But there are no spare parts for software. Any software error indicates error in design or coding of that software. Hence software maintenance involves more complexity than hardware maintenance.
  - Mostly software is custom built rather than assembled from existing components.
  - Computer hardware can be assembled from existing components as per our requirements. But that is not the case for software. Software is developed according to customer's need.

1.3.6 Software Crisis

- Most of software engineers refer the problems associated with software development as "Software crisis".
- The causes of software crisis are linked to all the problems and complexities associated with development process. Following are some most encountered problems associated with development process :
  1. Running out of time i.e. deadline crossed
  2. Over budget
  3. Software inefficient
  4. Low quality
  5. Not up to expectations of customers
  6. No enough development team

1.3.7 Legacy Software

- The term legacy software refers to older software developments that were poorly designed and documented. It had supported for many years. The legacy systems may or may not remain in use.
- Even if it is no longer used, it may continue to impact the organization due to its historical role. Historic data may not have been converted into the new system format and may exist within the new system.

Syllabus Topic : Software Process

1.4 Software Process

SPPU - May 12, May 13, May 16

University Questions

Q. What is software process ?
(May 2012, May 2013, 3 Marks)

Q. Explain software Engineering process framework activities. **(May 2016, 5 Marks)**

Review Questions

Q. What are various Umbrella Activities associated in the Development Process ?

Q. Describe Common Process Framework with the help of diagram.

- A software process can be illustrated by the following Fig. 1.4.1. In the Fig. 1.4.1, a common process framework is exhibited. This framework is established by dividing overall framework activities into small number of framework activities.
- And these small activities are applicable to the entire project irrespective of its size and complexity. A framework is a collection of task sets and these task sets consists of :
  - o Collection of small work tasks
  - o Project milestones, deliverable i.e. actual work product and
  - o Software quality assurance points

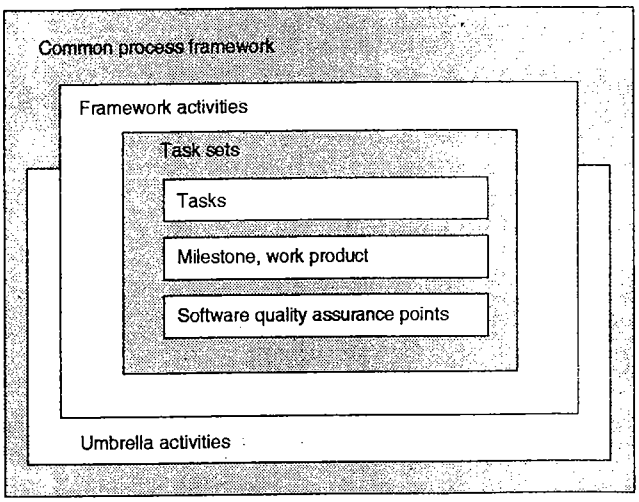


Fig. 1.4.1 : A software process

- There are various activities called **umbrella activities** are also there and these activities are associated throughout the development process. The umbrella activities include :

1. Software project tracking and control
2. Risk management
3. Software Quality Assurance (SQA)
4. Formal Technical Reviews (FTR)
5. Measurement
6. Software Configuration Management (SCM)
7. Reusability Management
8. Work product preparation and production

- All these umbrella activities actually constitute the process model. Also the umbrella activities are independent and occur throughout the process

1.4.1 Umbrella Activities

SPPU - May 12

University Question

Q. What are framework and umbrella activities ?
(May 2012, 3 Marks)

The framework described in generic view of software engineering (Fig. 1.4.1 : A software process) is complemented by number of Umbrella activities. Typical **umbrella activities** are :

1. Software project tracking and control

- Developing team has to assess project plan and compare with predefined schedule.
- If project plan doesn't match with predefined schedule, necessary actions are taken to maintain the schedule.

2. Risk management

Risk is event that may or may not occur. But if that event happens, it causes some unwanted outcomes. Hence proper management of risk is required.

3. Software Quality Assurance (SQA)

- SQA is nothing but planned and systematic pattern of activities those

are required to give guarantee of software quality.

- For example during software development, meetings are conducted in every stage of development to find out defects and suggest improvement to yield good quality software.

4. Formal Technical Reviews (FTR)

- FTR is a meeting conducted by technical staff. The purpose of meeting is to detect quality problems and suggest improvements.
- The technical staff focuses on quality from customer's point of view, i.e. what customer exactly wants.

5. Measurement

- It includes the efforts required to measure the software.
- Software can not be measured directly. It is measured by some direct measures (e.g. cost, lines of code, size of software etc) and indirect measures (e.g. quality of software, which is measured in terms of other factors. Hence it is an indirect measure of software.)

6. Software Configuration Management (SCM)

It manages the effects of change throughout the software process.

7. Reusability management

- It defines criteria for product reuse.
- If software components developed for certain application can be used in development of other applications, then it's good quality software.

8. Work product preparation and production

It includes the activities required to create documents, logs, forms, lists and user manuals for developed software.

Syllabus Topic : Software Engineering Practice

1.5 Software Engineering Practice

- Practice is collection of concepts, principles methods and tools those are considered while planning and developing the software.
- The software engineer implements this collection on daily basis for throughout development process of the software.
- Software engineers and managers apply the practice of software engineering.

Elements of Software Practice

Following are the elements of software practice those are applied irrespective of the process model chosen for the development.

- Concepts
- Principles
- Methods
- Tools

1.5.1 The Essence of Practice

George Polya has presented the concept "The Essence of Practice" in four sequential steps. These steps are :

- To understand the problem (Communication and Analysis)
- To plan the solution (Modeling and software design)
- To carry out the plan (Coding)
- To examine the result for accuracy (Testing and software quality assurance)

To understand the problem (Communication and analysis)

This goal is achieved by the basic activities 'Communication with Customer' and 'Requirement analysis'.

- Requirement analysis is a communication intensive activity. If there is

misinterpretation of customer requirements, then obviously, the requirement analysis will be wrong.

- Hence, wrong design will be formed and finally implementation will be wrong. So, customer will not get his expected results.
- Hence with 'customer communication' data, functions and behaviour of the system to be implemented must be stated clearly during 'requirement analysis' activity.
- In other words, practice is essential to understand problem statement.

To plan the solution (Modeling and software design)

This goal is achieved by activities Modeling and software design. Here following issues are considered.

Reusable components

To implement required data, functions and behaviour of the system, the 'reusable' components are searched. 'Reusable components' are the components those developed for some specific application and can be reused in development of other systems.

- **Subsystems :** The main system requirements are divided into different groups so that by implementing 'subsystems' modularized solution can be achieved. By combining all subsystems, the combined result is produced by main system.
- **Simple design :** The simple design is created so that design can be easily implemented using appropriate programming language.

To carry out the plan (Code generation)

- This goal is achieved in 'Coding' step. The design is translated into code by using appropriate programming language. The focus is given on following issues :
  - All design details must reflect into coding i.e. actual implementation.
  - Each module of coding must be correct to generate required output.

To examine the result for accuracy (Testing and quality assurance)

This goal is achieved 'Testing and Assurance' activities. This step focus on following issues :

- To check for expected results.
- To check whether required testing tools are available.
- To check for data, functions and behaviour of the system.
- To check whether system is maintaining good quality.
- To implement steps for software quality assurance.

1.5.2 Core Principles

The term 'principle' implies "some important law or assumption". In software engineering, the principles must be followed in every stage of software development.

For example

- The principles may be applied to 'software development' as whole.
- Principles those are applied to 'Communication' stage of software.
- Principles those are applied to 'Planning' stage of software.
- Principles those are applied to 'Analysis Modeling' stage of software.
- Principles those are applied to 'Design Modeling' stage of software.
- Principles those are applied to 'Construction' stage of software.
- Principles those are applied to 'Testing' stage of software
- Principles those are applied to 'Deployment' stage of software.

David Hooker's seven principles

David Hooker has proposed seven principles called as 'core principles'. These principles are applied to 'software development' as whole.

- **Principle Number 1 : The reason at all exists.**

  The complete software is one, which satisfies all the requirements by 'customer's point of view'. Hence before beginning a software project, be sure that the software has a business purpose. Hence first priority goes to customer's satisfaction.

- **Principle Number 2 : Keep simple but perfect.**

  The software design must be as simple as possible. If there are too many interfaces in the system, it's difficult to construct and implement such type of system. Hence design should be simple enough. But, simple design doesn't mean to skip the requirements. Hence design should be simple and perfect.

- **Principle Number 3 : Maintain the vision.**

  A clear vision is required for the success of the project. Hence maintain the architectural vision of software while designing it.

- **Principle Number 4 : What you produce, others will consume.**

  Other members in the team may use the software, which is analyzed, designed, and constructed by one person in a team. After deployment of software to the customer, the software 'enhancement' may be required as per customer's demand. Hence to maintain the software or to extend it, the software may be referred to develop other applications. Hence, the system must be transparent enough.

- **Principle Number 5 : Be Open to the future.**

  The solution for any problem must be generalized and not specific. The generalized solution can be reused for other applications. The technology changes day to day. Today's software gets outdated tomorrow. The programming languages used for software development may be outdated tomorrow. Hence the developed system must be generalized enough to absorb all these changes.

- **Principle Number 6 : Plan ahead for reuse.**

  'Reusability' is a property of good quality software. 'Reusable' software can be used in the development of other applications. Using Object Oriented programming languages like C++, it's possible to develop reusable modules. If reusable modules are available, then system can be developed very fast.

- **Principle Number 7 : Think for better solutions.**

  When we think about something, we get more ideas. If we are having solution and we think for better solution, the better idea will come out. This can improve the quality of system.

Syllabus Topic : Software Myths

1.6 Software Myths

SPPU - Dec. 16

University Question

Q. State and Explain different myths.

(Dec. 2016, 5 Marks)

Review Questions

Q. What is software myth ? Give an example. ?

Q. List and explain practitioners myths and what are their corresponding realities ?

Q. State a software myth each which customers, developers and project manager believe. What is the reality in each case ?

- Dictionary meaning of myth is **'fable stories'**. It is a fiction, imagination or it is a thing or story whose existence is not verifiable. In relation with computer software myth is nothing but the misinformation, misunderstandings, or confusions propagated in software development field.
- In software development the myths can be considered as three level myths.

1. Management level myths
2. Customer level myths
3. Practitioner level myths

1.6.1 Management Level Myths (or Manager Level Myths)

SPPU - May 12, Dec. 12, May 13

University Questions

Q. Explain in detail software myths: Management myths. **(May 2012, May 2013, 3 Marks)**

Q. What are the management myths? **(Dec. 2012, 4 Marks)**

➢ **Myth**

Manager thinks "there is no need to change approach to software development. We can develop same kind of software that we have developed ten years ago".

➢ **Reality**

- Application domain may be same but quality of software need to improve according to customer's demand.
- The customer demands change time to time.

➢ **Myth**

- We can buy software tools and use them.

➢ **Reality**

- Software tools are readymade software that helps in creation of other software e.g. Microsoft front page/ Microsoft Publisher. All these software can be used for web development.
- Rational rose used to draw UML diagrams (e.g. use cases, sequential, class diagrams etc).
- Such software tools are available for every stage of software development (Requirement analysis, designing, coding, testing etc). But majority of software developers do not use them. Moreover, these tools also have some limitations.

➢ **Myth**

Manager thinks "when needed, we can add more programmers for faster software development".

➢ **Reality**

- When new peoples are added to the project, the training must be given to such newcomers.
- Hence people previously working on that project spend time for training people.
- Hence project can not be completed fast just by adding more people at any time during project development.

➢ **Myth**

If the developer outsource the software project to a third party, he can be tension free and wait for the project to done smoothly.

➢ **Reality**

When an organization is unable to manage and control the software projects internally, it will also be very difficult for them to get the project done by outsourcing it.

➢ **Myth**

There is a book that contains standards and procedures for developing software, it will provide the developer everything that he needs in the development process.

➢ **Reality**

The rule book exists. But the questions arise :

- Is it actually used by the developers ?
- Are software developers aware of this type of book of standards and procedures ?
- Does it contain all the modern engineering practices ?
- Is it foolproof ?
- Does it focus on quality ?
- Does it streamlined to timely delivery ?
- In most of the cases, the simple answer to all these queries is a big "NO".

➢ **Myth**

- The organization has state-of-the-art development tools.
- They have the latest configuration computers for their developers to provide the good platform to produce their work efficiently.

➢ **Reality**

- In actual scenario the latest hardware configuration computer will not help in producing high-quality software.
- But the Computer-Aided Software Engineering (CASE) tools are very important for achieving good quality software.
- In most of the organizations the software developers do not these CASE tools effectively and efficiently.

1.6.2 Customer Level Myths

➢ **Myth**

Only the general statement is sufficient and no need to mention detail project requirements.

➢ **Reality**

- Only general statement is not sufficient for project development. Other detail project requirements are also essential.
- Customer must provide other information, design details, validation criteria.
- Hence during complete software development process customer and developer communication is essential.

➢ **Myth**

Project requirements continuously change but can be easily accommodated in software.

➢ **Reality**

- If change is requested in early stages of software development, it can be easily carried out. But if change is requested in later stages, cost of change rapidly increases.
- If change is requested in maintenance, modifications are required in design, coding, testing.
- Hence cost of modification rapidly increases. Thus changes can't be easily absorbed. They result in increase in cost.

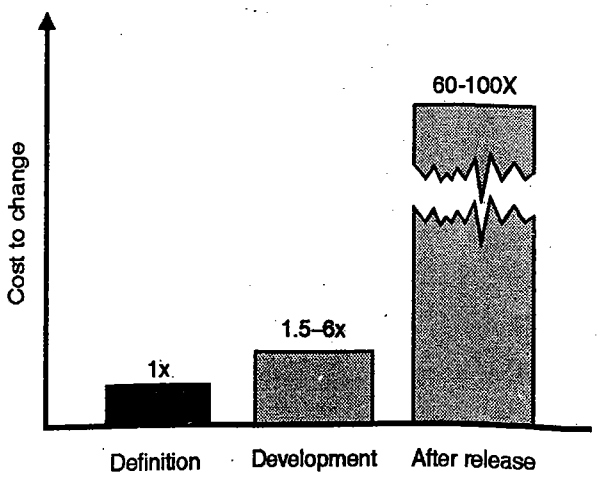


Fig. 1.6.1: The impact of change

1.6.3 Practitioner Level Myths (or Developer Level Myths)

SPPU - May 12, May 13, Dec. 15

University Questions

Q. Explain in detail software myths: Practitioner's myths. **(May 2012, May 2013, 4 Marks)**

Q. Discuss practitioner's myths of software development. **(Dec. 2015, 5 Marks)**

➢ **Myth**

Practitioners think "Once we write program and get into work, our job is over".

➢ **Reality**

Almost 60 to 80 percent work is required after delivering the project to the customer for the first time.

➢ **Myth**

Until I get program running, I have no way of assessing its quality.

➢ **Reality**

- The most effective quality assurance mechanisms can be applied during project development. The formal Technical Reviews are conducted to assure the quality
- Formal Technical Review is meeting conducted by technical staff. FTR is applied in any stage of software development (Requirement analysis, designing, coding, testing etc).
- FTR is not problem solving activity but it is applied to find out defects in any stage of software development. These defects can then removed to improve the quality of software.

➢ **Myth**

When project is successful, deliverable product is only working program.

➢ **Reality**

Working program is just part of software product. Actual project delivery includes all documentations, help files and guidance for handling the software by user.

➢ **Myth**

The software engineering process creates larger and unnecessary documentation and ultimately it will slow down the process.

➢ **Reality**

- Software engineering the process that creates the quality and it is the not the process of only creating the documents.
- The good quality of the product will reduce the extra work involved in rework.
- And finally reducing the overhead of rework will lead to quicker development to complete the desired work on scheduled time.

❑❑❑

CHAPTER 2

UNIT I

Process Models

Syllabus

Generic Process model, Analysis and comparison of Process Models: Waterfall Model, Incremental Models, Evolutionary Models, Concurrent, Specialized Process Models, Personal and Team Process Models, Introduction to Clean Room Software Engineering.

Syllabus Topic : Generic Process Model

2.1 Generic Process Model (or Generic Process Framework)

SPPU - May 16, Dec. 16

University Questions

Q. What is generic process model ? Explain its activities. **(May 2016, 5 Marks)**

Q. Write short note on Generic process model. **(Dec. 2016, 5 Marks)**

Review Question

Q. What is generic process model ? Explain its activities.

A software process is collection of various activities. There are five generic process framework activities:

1. Communication
2. Planning
3. Modeling
4. Construction
5. Deployment

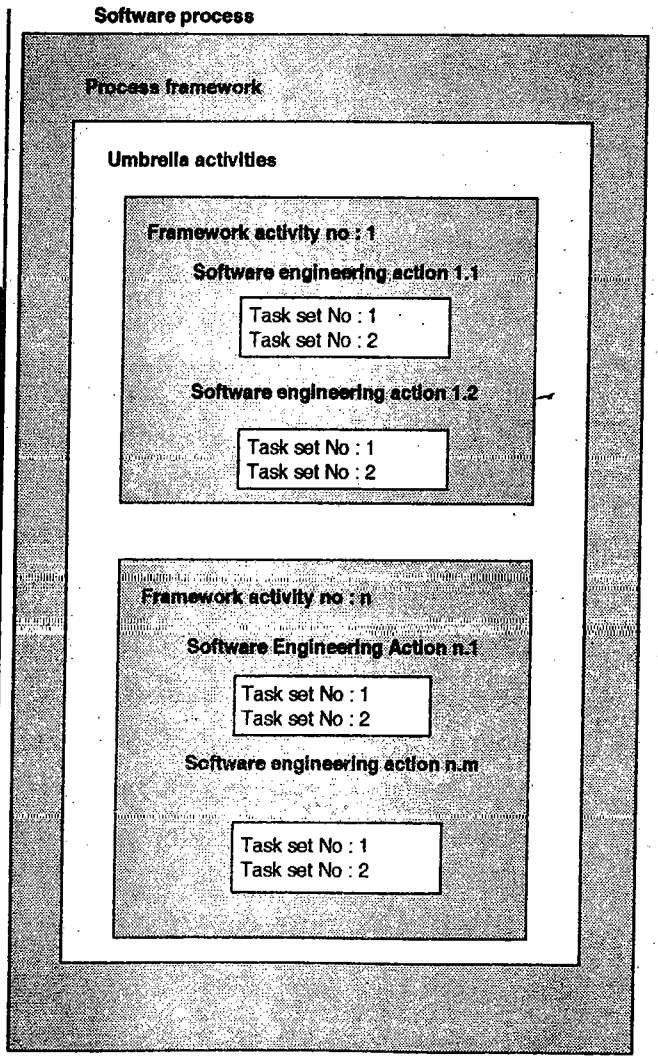


Fig. 2.1.1 : A software process framework

2.1.1 Communication

- The software development process starts with communication between customer and developer.

- According to waterfall model customer must state all requirements at the beginning of project.

2.1.2 Planning

It includes complete estimation (e.g. cost estimation of project) and scheduling (complete timeline chart for project development) and tracking.

2.1.3 Modeling

- It includes detail requirement analysis and project design (algorithm, flowchart etc).
- Flowchart shows complete pictorial flow of program whereas algorithm is step-by-step solution of problem.

2.1.4 Construction

It includes coding and testing steps :

(i) **Coding :** Design details are implemented using appropriate programming language.

(ii) **Testing :** Testing is carried out for the following reasons :

- To check whether the flow of coding is correct or not.
- To check out the errors of program e.g. in C program just by pressing F7 key we check step by step execution of program or by using "Add-Watch" we add the variables and watch the values of variables.
- To check whether program is giving expected output as per input specifications.

2.1.5 Deployment

- It includes software delivery, support and feedback from customer.
- If customer suggest some corrections, or demands additional capabilities, then changes are required for such corrections or enhancement.
- These five generic framework activities can be used for :
  - o Development of small programs
  - o Creation of large programs
  - o Development of large system based programs
- In addition to these five generic framework activities, various umbrella activities are also associated throughout the development process.

Syllabus Topic : Analysis and Comparison of Process Models

2.2 Prescriptive Process Models

- All the software organizations describe a unique set of framework activities for their own development processes. In general, it should adopt all the generic framework activities and the set of tasks defined in earlier Section 2.1 i.e. generic process model. Whatever process model is chosen by the organization, but it should encompass the following framework activities :
  1. Communication
  2. Planning
  3. Modeling
  4. Construction
  5. Deployment
- The name 'prescriptive' is given since the model prescribes set of activities, actions, tasks, quality assurance and change control mechanism for every project. Each of the prescriptive models also prescribes a workflow.
- Workflow is defined as the flow of process elements and the manner in which they are interrelated. All the generic framework activities are defined earlier, but each of the prescriptive models put different

emphasis to these generic framework activities and give different workflow.

- In the following section, we describe some of the prescriptive process models :

1. The waterfall model
2. Incremental process models
3. Evolutionary process models
4. The specialized process models

Syllabus Topic : Waterfall Model

2.2.1 Waterfall Model

SPPU - May 12, Feb. 16

University Question

Q. Explain in detail all the process phases of waterfall process model and state merits/demerits of the same. **(May 2012, 3 Marks)**

Review Questions

Q. What are the disadvantages of the water fall model ?

Q. Explain the waterfall model with work products of each activity.

Q. Explain advantages and disadvantages of waterfall model.

- This model is also called as **'Linear sequential model'** or **'Classic life cycle model'**. Classic life cycle paradigm begin at system level and goes through analysis, design, coding, testing and maintenance.

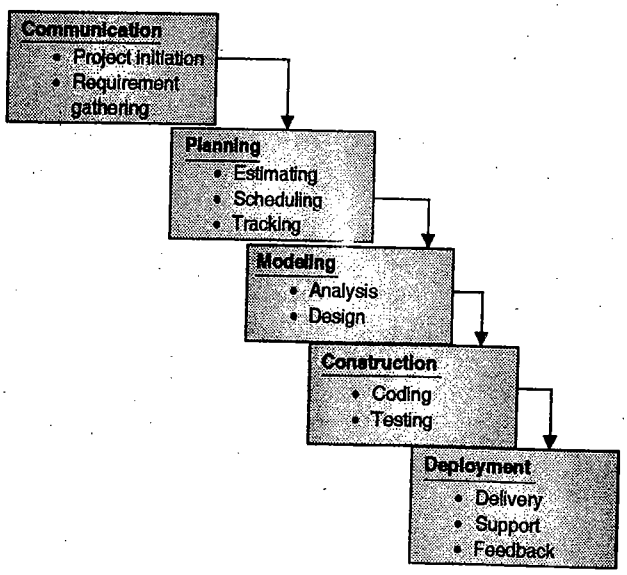


Fig. 2.2.1 : The Waterfall model

- An alternative design for 'Linear Sequential Model' is as follows :

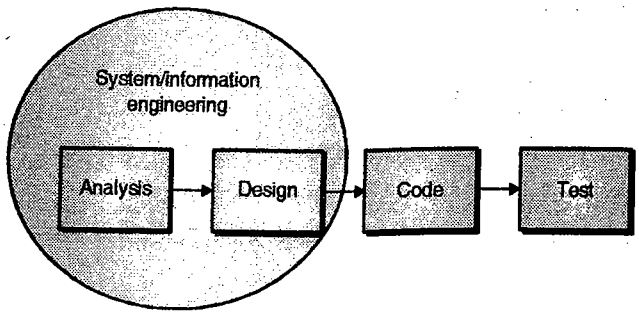


Fig. 2.2.2 : The linear sequential model

1. Communication

- The software development process starts with communication between customer and developer.
- According to waterfall model customer must state all requirements at the beginning of project.

2. Planning

It includes complete estimation (e.g. cost estimation of project) and scheduling (complete timeline chart for project development) and tracking.

3. Modeling

- It includes detail requirement analysis and project design (algorithm, flowchart etc).
- Flowchart shows complete pictorial flow of program whereas algorithm is step-by-step solution of problem.

4. Construction

It includes coding and testing steps :

(i) Coding : Design details are implemented using appropriate programming language.

(ii) Testing : Testing is carried out to check whether flow of coding is correct, to check out the errors of program e.g. in C program just by pressing F7 key we check step by step execution of program or by using "Add-Watch" we add the variables and watch the values

of variables, to check whether program is giving expected output as per input specifications.

5. **Deployment**

   - It includes software delivery, support and feedback from customer.
   - If customer suggest some corrections, or demands additional capabilities then changes are required for such corrections or enhancement.

Merits / Advantages

1. This model is very simple and easy to understand and use.
2. Waterfall model is systematic sequential approach for software development. Hence it is most widely used paradigm for software development.
3. In this approach, each phase is processed and completed at one time and thus avoids phases overlapping.
4. It is very easy to manage since all the requirements are very well understood in the beginning itself.
5. It establishes the milestones whenever the products are produced or reviews available.

Demerits / Disadvantages

1. Problems in this model remain uncovered until software testing.
2. One of the disadvantages of this approach is 'blocking states'. In blocking state, the members of development team waits for other members to complete the dependent tasks. Due to this, huge amount of time spent in waiting rather than spending time on some productive work. In today's world, the software work is fast paced and thus waterfall model is not useful.
3. According to this model customer must state all his requirements at the beginning stage of development, which is difficult for the customer.
4. This model is step-by-step systematic sequential approach. Ultimately, customer gets the working version of software too late. Hence customer requires patience.
5. This approach is actually not realistic and does not match with real projects.
6. Also it does not incorporate the risk assessment.
7. Finally it is useful for smaller projects only where requirements are well understood in the beginning only.

Syllabus Topic : Incremental Models

2.2.2 Incremental Models

SPPU - May 12

University Question

Q. Explain in detail the model: Incremental model. **(May 2012, 4 Marks)**

- Using these models a limited set of customer's requirements are implemented quickly and are delivered to customer. Then modified and expanded requirements are implemented step by step.
- Actually in this approach, the overall functionalities are split to smaller modules and these modules are quickly developed and delivered. Then refinements are possible in later increments or versions.
- In this approach, the initial requirements are very well defined. Each increment is passed through the overall development cycle i.e. phases of classic life cycle discussed earlier.
- The customer's feedback is very important after each of the increments (or releases) and next increment is developed. This process continues till the product is finished i.e. all the requirements are satisfied.
- Following are the two types of incremental process models :

1. The incremental model
2. The RAD model

Syllabus Topic : The Incremental Model

2.2.2.1 The Incremental Model SPPU - Dec. 16

University Question

Q. Explain the incremental software process model in detail. **(Dec. 2016, 5 Marks)**

Review Question

Q. Explain the incremental model with advantages and disadvantages ?

- The incremental model combines the elements of waterfall model applied in an interactive fashion.
- The first increment is generally a core product. Each increment produces the product, which is submitted to the customer. The customer suggests some modifications.

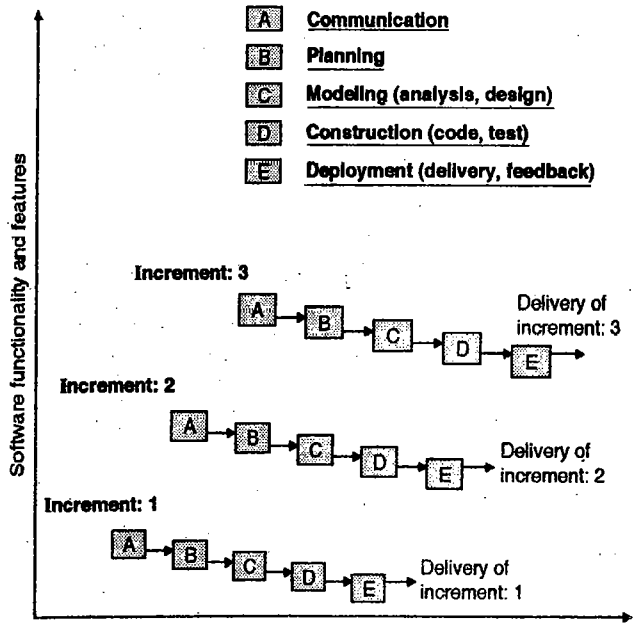


Fig. 2.2.3 : Project calendar time

- The next increment implements customer's suggestions and some additional requirements in previous increment. The process is repeated until the product is finished.
- Consider the development of word processing software (like MS Word or Word Star) using incremental model.

Increment – 1

Basic file management functions like : New, Open, Save, Save as etc. can be implemented in first increment.

Increment – 2

- More sophisticated editing and document production capabilities can be implemented in second increment.
- For example rulers, margins and multiple font selection can be implemented.

Increment - 3 Spelling, grammar checking and auto correction of words is implemented in third increment.

Increment - 4

The final advanced page layout capabilities are developed in fourth increment. And so on till the product is finished.

Merits / Advantages

1. The incremental model is useful when the size of development team is small in the beginning or some team members are not available. Thus early increments can be implemented with small size of team.
2. If the product satisfies the client, then the size of team can be increased.
3. Also increments can be properly planned to handle all types of technical risks. For example some application may require a new hardware that is presently not available. In this situation the increments can be planned to avoid the use of that hardware.
4. The initial product delivery is faster and cost of development is low.
5. The customers can respond to the functionalities after each increment and come up with feedback.

Demerits / Disadvantages

1. The cost of finished product may be increased in the end beyond the cost estimated.
2. After each increment, the customer can

demand for additional functionalities that may cause serious problems to the system architecture.

3. It needs a very clear and complete planning and design before the whole system is broken into small increments.

Syllabus Topic : The RAD Model

2.2.2.2 The RAD Model

SPPU - May 14

University Question

Q. Explain RAD model. **(May 2014, 4 Marks)**

Review Question

Q. Explain the RAD model with advantages and disadvantages?

- RAD (Rapid Action Development) model is high-speed adaptation of waterfall model. Using RAD model, software product can be developed within a very short period of time i.e. almost within 60 to 90 days.
- The initial activity starts with communication between customer and developer. Depending upon initial requirements the project planning is done. Now the requirements are divided into different groups and each requirement group is assigned to different teams.
- Like other approaches, in RAD approach also all the generic framework activities are carried out. These generic framework activities are already discussed earlier.
- When all teams are ready with their final products, the product of each team is integrated i.e. combined to form a product as a whole.
- In RAD model each team carries out the following steps :

1. **Communication :** It understands the business problem and information for software.
2. **Planning :** Since various teams work together on different modules simultaneously, planning is important part of development process.
3. **Modeling :** It includes :
   - (i) **Business Modeling :** It includes information flow among different functions in the project e.g., what information will be produced by each function, which functions handles that information etc.
   - (ii) **Data Modeling :** It includes different data objects used in software and relationship among different objects.
   - (iii) **Process Modeling :** During process modeling, process descriptions are created. e.g. add, modify, delete etc.
4. **Construction :** It includes :
   - (i) **Component reuse**

     Reusable components means the software components i.e. modules or procedures that are developed for specific application can be reused in development of other application.
   - (ii) **Automatic code generation**

     The software producing the codes after providing proper inputs or selecting readymade options, e.g. Microsoft FrontPage used in Web development, Rational Rose used for Object Oriented analysis and designing.
   - (iii) **Testing**

     Testing is carried out to check whether flow of coding is correct, to check out the errors of program e.g. In C program just by pressing F7 key we check step by step execution of program or by using "Add-Watch" we add the variables and watch the values of variables, or check whether program is giving expected output as per input specifications.

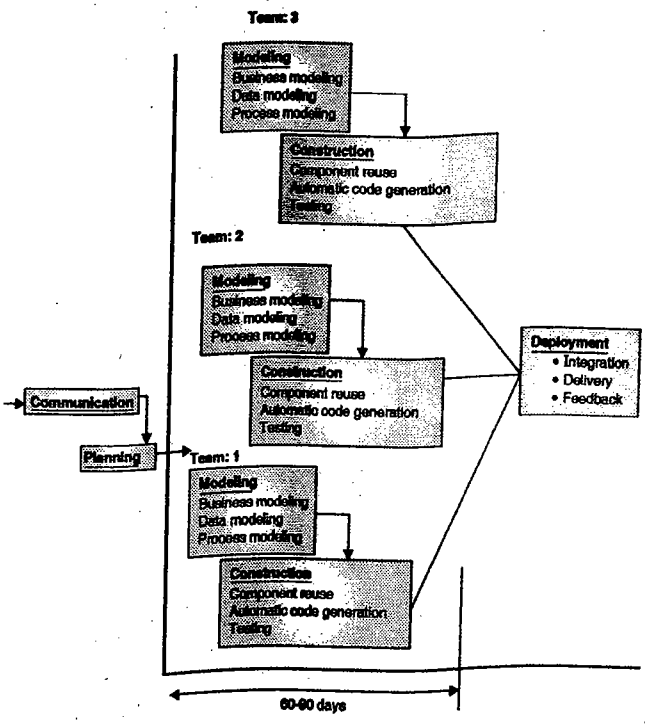


Fig. 2.2.4 : The RAD Model

Merits / Advantages

1. Using the RAD approach, a product can be developed within very short period of time.
2. It supports increased reusability of the components.
3. It results in minimal code writing as it supports automatic code generation.
4. It encourages the customer feedback.
5. In this approach, quick initial reviews are possible.
6. In this approach, modules integration is done from the beginning thus resolving number of integration issues.

Demerits / Disadvantages

1. It can be used, if sufficient number of staff is available. Thus, it requires sufficient man power to create number of teams.
2. In RAD approach, many teams work parallel to implement different functions of same project. Hence all teams must work with equal speed. If one team lags behind the schedule, overall project delivery will be late.
3. If system can not be modularized properly, then building the components for RAD model will be problematic.
4. The RAD model is not appropriate when technical risks are high. (e.g. a new application makes heavy use of new technology).
5. This approach is useful for only larger projects.
6. The RAD model had two primary disadvantages as follows :
   - **Reduced scalability :** It occurs because a RAD application evolves from a prototype to a finished application. Thus there is less scope for scalability.
   - **Reduced features :** It occurs due to time boxing. Time boxing is a time management technique in which the task is to be completed in a given frame of time called time boxes. Thus the features are pushed to be completed to the later releases due to short amount of time.

Syllabus Topic : Evolutionary Models

2.2.3 Evolutionary Models

Review Question

Q. What do you mean by evolutionary process models ? Explain spiral model as an evolutionary process model.

- Evolutionary process models are iterative type models. Using these models the developer can develop increasingly more complete versions of the software.
- As the requirements change very often, the end product may be unrealistic and a complete version of the product is not possible. To overcome this drawback, the

concept of limited version product is introduced and this version is gradually completed over the period of time.

- Each version is refined based on the feedbacks till it is completed.
- Following are the examples of evolutionary process models :

1. The Prototyping paradigm
2. The Spiral model
3. The Concurrent development model

2.2.3.1 The Prototyping Paradigm

SPPU - May 12

University Question

Q. Explain in detail the model: Prototyping model. **(May 2012, 4 Marks)**

Review Questions

Q. What do you mean by working and throwaway prototypes? Explain how they are used in prototyping model.

Q. Explain advantages and disadvantages of prototyping model.

- The **prototyping paradigm** offers best approach for human machine interaction. Ideally prototype serves as a mechanism for identifying software requirements.

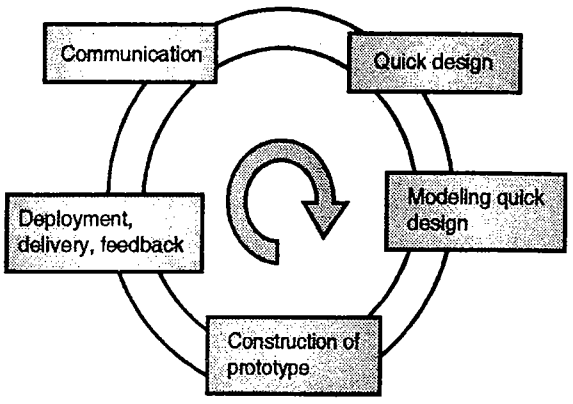


Fig. 2.2.5 : The prototyping model

- The prototyping approach is often called as **throwaway prototyping**. By using this approach, there is a rough demonstration of the customer's requirements i.e. the prototype used in demonstration is just a throwaway prototype.
- If working prototype is built, developer can use software tools if required (e.g. report generators).
- This produces working program quickly. Thus prototype can be served as 'first system'.
- Different phases of prototyping model are :

1. Communication

- The software prototyping paradigm starts with the communication between the developer and the customers.
- The software engineer and the customer meet regularly to define the overall objectives of the desired software and to identify it's requirements.

2. Quick design

- Quick design focuses on those aspects of software that are visible to the customer.
- It includes clear input, output formats and human machine interfaces.

3. Modeling quick design

The model of software is now built that gives clear idea of the software to be developed. This enables the developer to better understand the exact requirements.

4. Construction of prototype

The concept of modeling the quick design leads to the development of prototype. This construction of prototype is deployed by the customer and it is evaluated by the customer itself.

5. Deployment, Delivery, Feedback

- As picture of software to be built is very clear, customer can give his suggestions as a feedback.

- If result is satisfactory, the model is implemented otherwise process is repeated to satisfy customers' all requirements.

Merit / Advantage

Prototyping makes requirements more clear and system more transparent.

Demerits / Disadvantages

- The software developer generally compromises in the quality to get the working prototype quickly.
- Due to this reason, sometimes inappropriate operating system or programming language may be selected. He may use and implement inefficient algorithm.

2.2.3.2 The Spiral Model

SPPU - Dec. 12, May 13, May 16

University Questions

Q. Explain in detail Spiral model with its merits/demerits. **(Dec. 2012, 8 Marks)**

Q. Explain Spiral model in detail. **(May 2013, 4 Marks)**

Q. Explain Spiral model. Give an example of application using spiral model. **(May 2016, 5 Marks)**

Review Question

Q. What are the advantages and disadvantages of spiral model ?

- The **Spiral model** is combination of well known waterfall model and iterative prototyping. It yields rapid development of more complete version of software.
- Using spiral model software is developed as series of evolutionary releases. During the initial releases, it may be just paperwork or prototype. But during later releases the version goes towards more completed stage.
- The spiral model can be adopted to apply throughout the entire lifecycle of the application from concept development to maintenance. The spiral model is divided into set of framework activities defined by software engineer team.
- Each framework activity represents one segment of spiral as shown in Fig. 2.2.6.

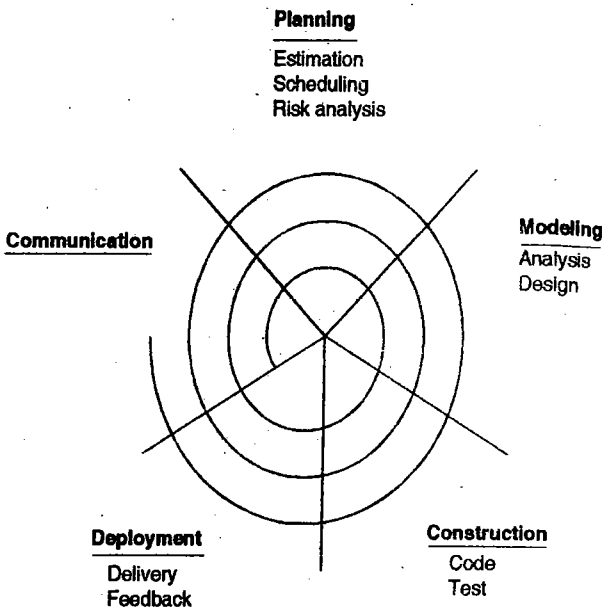


Fig. 2.2.6 : The Spiral Model

- The initial activity is shown from centre of circle and developed in clockwise direction. Each spiral of the model includes following steps :

1. Communication

The software development process starts with communication between customer and developer.

2. Planning

It includes complete estimation (e.g. cost estimation of project) and scheduling (complete timeline chart for project development) and risk analysis.

3. Modelling

- It includes detail requirement analysis and project design (algorithm, flowchart etc).
- Flowchart shows complete pictorial flow of program whereas algorithm is step by step solution of problem.

4. Construction

It includes coding and testing steps :

(i) **Coding** : Design details are implemented using appropriate programming language.

(ii) Testing : Testing is carried out.

5. Deployment

- It includes software delivery, support and feedback from customer. If customer suggest some corrections, or demands additional capabilities then changes are required for such corrections or enhancement.
- Note that after customer evaluation, next spiral implements, 'customer's suggestions' plus 'enhancement plan'. Thus, each of iteration around the spiral leads to more completed version of software.

Merits / Advantages

1. In this approach, project monitoring is very easy and more effective compared to other models.
2. It reduces the number of risk in software development before they become serious problems.
3. It is suitable for very high risk projects.
4. Project estimates i.e. schedule and cost is more realistic.
5. Risk management is in-built feature of spiral model.
6. Changes can be accommodated in the later stages of development.

Demerits / Disadvantages

1. If major risk is not discovered in early iteration of spiral, it may become a major risk in later stages.
2. Each iteration around the spiral leads to more completed version of software. But it's difficult to convince (especially in contract situation) to the customer that the model is controllable.
3. Cost of this approach is usually high.
4. It is not suitable for low risk projects.
5. Rules and protocols must be followed very strictly to implement the approach.

2.2.3.4 Differentiation between Prescriptive and Evolutionary Process Models

Review Question

Q. Differentiate between prescriptive and evolutionary process models.

Sr. No.	Prescriptive process models	Evolutionary process models
1.	Developed to bring order and structure to the software development process.	Evolutionary software processes do not establish the maximum speed of the evolution. Due to this development process becomes slow.
2.	Defines a distinct set of activities, actions, tasks, milestones, and work products that are required to engineer high-quality software	Evolutionary process models lack flexibility, extensibility, and high quality.
3.	It is more popular.	It is less popular.
4.	It provides complete and full developed systems.	Time does not allow a full and complete system to be developed.
5.	**Example** : Water fall model, Incremental models.	**Example** : Prototyping, Spiral and Concurrent models.

2.2.4 The Specialized Process Models

- Specialized process models have many characteristics of one or more of the conventional models described in earlier section.
- Specialized models are applied when a narrowly defined engineering approach is chosen.
- Following are some specialized process models :

1. Component-based development models
2. The formal methods model
3. Aspect-oriented software development

Specialized process models are covered in Section 2.4.

Syllabus Topic : Concurrent Model

2.3 Concurrent Model

Review Question

Q. Explain concurrent development model with advantages and disadvantages.

- The **concurrent development** model is also called as **concurrent model.** This model is more appropriate for system engineering projects where different engineering teams are involved. In system engineering stage, the project requirements are considered at system level.
- Diagrammatically it can be represented as a series of framework activities, task, actions and their associated states.
- Following Fig. 2.3.1 exhibits one element of the concurrent process model :

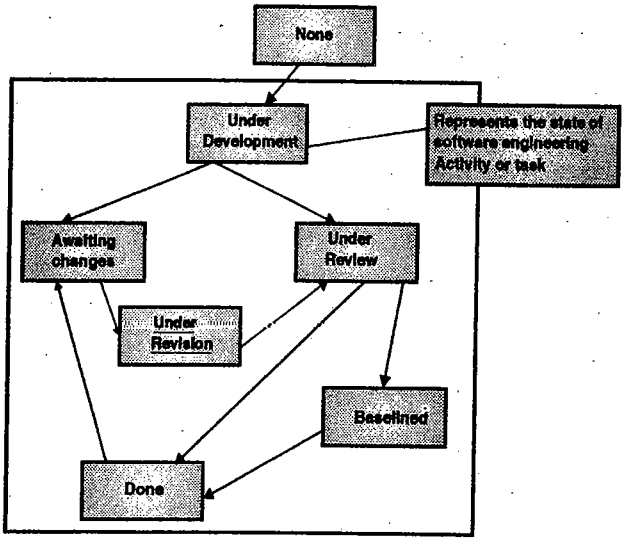


Fig. 2.3.1 : One element of concurrent process model (i.e. Modeling activity)

- The **modelling activity** is one of the states and other activities like **communication** or **construction** can also be represented in an analogous manner.
- Let the **communication activity** has completed its first iteration and available in **awaiting changes state.** The **modelling activity** has completed its initial communication and ready to move to **under development state** from none state.
- During these transitions, if customer indicates some modification in the requirement, then the **modelling activity** transits to **awaiting changes state** from **under development state.**
- Instead of considering activities, actions and tasks as sequence of events, it defines network of activities.
- The concurrent process model activities transits from one state to another state and this model is used in all types of software development and it provides very clear idea of the current state of the project.

Merits / Advantages

1. The concurrent development model is applicable to all types of software development processes.
2. It gives very clear picture of current state of the project.
3. It is easy to use and easy to understand.
4. This approach is flexible and number of releases determined by the development team.
5. New functionalities as elicited by the client can be accommodated late in the project.
6. It has immediate feedback from testing.

Demerits / Disadvantages

1. Since all the stages in the concurrent development model works concurrently, any change in the requirement from the client may halt the progress. This may happen due to dependent components among different stages and it lead to more longer development cycles as compared the planned cycles.
2. It requires excellent and updated communication between the team members. This may not be achieved all the time.
3. The SRS must be updated at regular intervals to reflect the changes.

Syllabus Topic : Specialized Process Models

2.4 Specialized Process Models

- Special process has many characteristics of one or more of the conventional models presented in previous section.
- Specialized models are applied when a narrowly defined engineering approach is chosen.

1. Component-based development models
2. The formal methods model
3. Aspect-oriented software development

2.4.1 Component Based Development

- Component Based Development has many characteristics of spiral model. It is evolutionary in nature and includes iterative approach for software development.
- This model composes the applications from pre-packaged software components i.e. from existing software modules.
- Modeling and construction activity begin with identification of candidate components. These components can designed as either conventional software modules or object oriented classes or packages.
- The component is nearly independent and replaceable part of system.
- A package of classes is a collection of objects that work together to achieve some result.

Component Based Development includes following steps :

- Available components-based products are researched and evaluated for application domain.
- Component integration issues are considered.
- Software architecture is designed to accommodate the components.
- Components are integrated i.e. combined into architecture.
- Testing is carried out.
- Important feature of Component Based Development is that this model leads to 'software reuse'.
- Reusable components means the components i.e. software modules those are developed for specific application can be reused in development of other application projects. Reusability provides many benefits for software development.

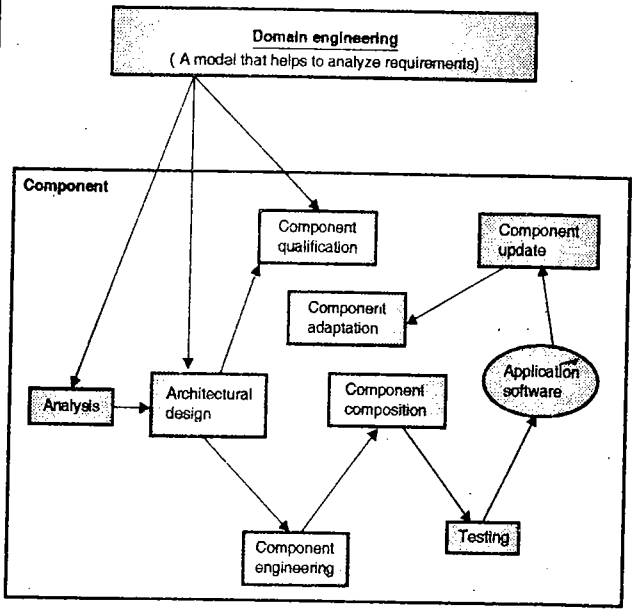


Fig. 2.4.1

Component Based Software Engineering (CBSE) includes following steps :

- 'Domain Engineering' creates a model of application domain that is used as a basis of analyzing the requirements.
- Software architecture provides inputs for design of the application.
- Finally after reusable components are purchased, selected from existing library, or constructed as a part of domain engineering, they are made available to software engineering during component-based development.

Merits / Advantages

1. The component based development model supports reusability.
2. Due to reusability, there is reduction in development cycle time.
3. Ultimately the cost of overall development reduces substantially.
4. There is significant increase in productivity.

Demerits / Disadvantages

1. The component based development model suffers from several problems. One of the serious problems is late discovery of the errors due to component interface or architectural mismatches.
2. The problem in encapsulation occurs due to functional overlapping.
3. It is very difficult to find that in which component a particular function will be implemented.
4. It is very difficult to achieve a complete separation of process from component.
5. This approach can not be fully utilized if a development organization does not adopt the principles of component based software engineering.

2.4.2 The Formal Methods Model

- The formal methods model includes set of activities that leads to formal mathematical specification of computer software. Using formal methods, a software engineer can specify, develop and verify computer based system by applying mathematical notations.
- Following are some examples of formal methods model :

A Symbol table

- A program is used to maintain a symbol table. Such tables can be used in many different types of applications. For example, a table that includes a collection of items without any duplication.
- A typical example is names of users of a system without duplicate entries. If such table is examined during execution of the system any time, it will never show duplicate names for use 'Saina'.

  1. Tendulkar 2. Dhoni
  3. Sehwag 4. Yuvraj
  5. Raina

- (An example symbol table of users used in formal methods model)

Merits / Advantages

1. When formal methods are used, many problems get eliminated those are difficult to overcome using other software engineering paradigms. For example Ambiguity, incompleteness and inconsistency can be discovered and corrected more easily using mathematical analysis.
2. The formal methods used in the design process provide the basis for program verification. They also enable the software developer to uncover the undetected errors and then fix them.
3. Using this method it's possible to develop accurate and defect free software.

Demerits / Disadvantages

1. This model is just time consuming and more expensive with compared to other methods.
2. As formal method models requires that software developers must have knowledge about formal methods. Otherwise training about formal methods must be given to developers.
3. It is difficult to use the model as a communication mechanism for technically unsophisticated customers.

Problems with formal methods model

- This model is just time consuming and more expensive with compared to other methods.
- As formal method models requires that software developers must have knowledge about formal methods. Otherwise training about formal methods must be given to developers.
- It is difficult to use the model as a communication mechanism for technically unsophisticated customers.

Where the models are essential ?

In the development of critical software (for example developers of aircraft and medical devices), even a minor calculation mistake can cause serious problem, format methods models are essential.

2.4.3 Aspect-Oriented Software Development

- Irrespective of the software process that is chosen, the builder of complex software implements a set of localized features, functions and information content. These localized software characteristics are models as components (e.g. Object Oriented Classes) and then constructed within the context of system architecture.
- As modern computer based systems become more sophisticated, certain "concerns" (i.e. customer required properties of technical interest-span the entire architecture). Some concerns are high-level properties of the system (e.g. security). Other concern affects functions (e.g. application of business rules), while others are systematic (e.g. task synchronization or memory management).
- In most of the large systems the relationship between different program components and their requirements are complex. One single requirement can be implemented by various components and vice versa that means one single component may be involved in various requirements.
- To address this particular problem and make programs easier and maintain good reusability, Aspect Oriented Software Engineering (AOSE) is used.
- AOSE implements system functionality at various places in the program. An important characteristic of AOSE is that it includes a definition of an aspect for its use in a proper place in a program.
- The important benefit of this approach is that it supports the separation of concerns into independent elements. For example, user authentication can be represented as an aspect that requests a login name and a password. This login name and password can be used in a program wherever authentication is required.
- The aspect oriented process is likely to adopt characteristics of spiral models and concurrent process models. The spiral model's evolutionary nature is best the aspects since in these models, aspects are identified and after that they are constructed.
- The concurrent development model has parallel nature and it is necessary, because aspects are developed independently by using local software components. Still aspects have a direct impact on these software components.

2.4.4 Need of Process Models

Workflow is defined as the flow of process elements and the manner in which they are interrelated. All the generic framework activities are defined earlier, but each of the prescriptive models put different emphasis to these generic framework activities and gives different workflow. This is the reason why

different process models are required in software development. Each model has its own importance and significance.

Syllabus Topic : Personal and Process Models

2.5 Personal and Process Models

- Two types of process models are possible for software development :
  - Personal Software Process (PSP)
  - Team Software Process (TSP)
- In PSP the process model, which is best, fitted for individuals in development team is selected. At the same time the model is convenient for whole team and organization.
- In TSP the model that is best fitted for team is selected. At the same time model also satisfies needs of individuals in team.

2.5.1 Personal Software Process (PSP)

- In this type each developer uses some process to build software.
- The process may be ad-hoc, inefficient, non effective and daily changing. Here each individual in the team requires training and careful instrumentation.
- This method implies personal measurement of the product. Individual person in team is responsible for project planning (e.g. estimating and scheduling).
- PSP model defines following five framework activities :

Planning

It includes planning of resources, scheduling etc.

High level design

The specifications for each component are identified and component design is created.

High level design review

Review implies reconsideration. The review of previous design is conducted to detect errors in design.

Development

It includes code generation, compilation and software testing.

Postmortem

The effectiveness of process is determined. Modifications may be applied for improvement of software.

Disadvantages

- PSP stresses for each software engineer to detect the errors and to develop individually the ideas to eliminate the errors.
- Here each individual in the team requires training. Training is relatively lengthy and training cost are high.
- The required software measurement is difficult for many software people.

2.5.2 Team Software Process (TSP)

- The goal of TSP is to build "self directed" project team that organizes itself to produce high quality software. The objectives of Team Software Process are :
  - To build self directed team that is capable to establish goals and plan the resources and schedules.
  - To assign managers to direct the team during the entire process of development.
  - Provide improvement guidance for the team.
  - Facilitate standards for teamwork.
- TSP defines following framework activities :
  - Launch
  - High-level design
  - Implementation
  - Integration and test
  - Postmortem

- TSP uses variety of scripts, forms and standards to guide the team members for their work.
- In TSP each project is launched using a sequence of tasks. This sequence of tasks is called launch script.
- The launch script includes following steps :
  - Reconsideration of objectives and planning of goals
  - Define overall team roles.
  - Define the team's development process including individual's role in team.
  - Deciding quality plan
  - Deciding the required support facilities.
  - Produce an overall development strategy.
  - Produce development plan for the entire project.
  - Decide detailed role of each member in the team.
  - Integrate plan of each member in team to produce complete plan.
  - Load distribution to all members to achieve a minimum overall schedule.
  - Risk assessment and tricks to avoid risks.

Syllabus Topic : Introduction to Clean Room Software Engineering

2.6 Introduction to Cleanroom Software Engineering

- Cleanroom software engineering is defined as an engineering approach that builds correctness in software being developed. Unlike the traditional approach of analysis, design, code, test and debug, it offers some different methodology.
- The main concept behind cleanroom software engineering is removing the dependency on costly processes i.e. defect removal processes by writing the correct code right from first line and verify its correctness before the testing phase starts.
- Cleanroom software engineering incorporates the quality approach of writing the code right from the beginning of the system and finally it accumulates into a complete system.
- We can say that cleanroom software engineering approach takes the software development process to a next level.
- In traditional development approach, if software fails, then we can face so many hazards that can challenge the human safety also. Due to software failure, we can face economic loss in business. Since the software is being used in nearly all the business worldwide e.g. banking sector, the success of banking sector totally depends on its correct software application in all the ways.
- Cleanroom software engineering, we can guarantee that it will not lead to any hazards like our traditional system explained in the above paragraph in case of failure of software.

2.6.1 Cleanroom Design

- Cleanroom software engineering in hardware fabrication technologies is cost effective and time effective and it prevents the introduction of any product defects.
- Rather than making a product and then remove the defects in its working, cleanroom software engineering approach focuses on systematic and disciplined engineering process that removes all possible errors and defects in early stages like specification and design and confirms a clean development.
- Three researches namely Mills, Dyer, and Linger in the area of software engineering first proposed the concept of cleanroom software engineering. They showed the systematic and disciplined approach to the

development process, but gained no popularity because people thought it is not practical and feasible.

- The another researcher explained three possible reason for failure of cleanroom software engineering :
  - This approach is very much theoretical and mathematical for the use in real software development process.
  - It offers no testing, because it is assumed that cleanroom software engineering approach is a disciplined approach of writing correct code and these codes are verified by using quality control. It deviates the from the real software process used today.
  - The cleanroom software engineering requires some rigorous processes in its development stages. Since most of the industry is still working on ad hoc level as discussed in capability maturity model proposed by Software Engineering Institute (SEI).
- Even though the above reasons exist in practical scenario, the investors still think of investing in cleanroom software engineering and that is its potential benefits.

2.6.2 Cleanroom Strategy

- The cleanroom software engineering approach uses the specific versions of all software engineering models. It uses the specialized version of incremental model as discussed in earlier.
- All the increments are developed by small independent teams or units. Each increment is certified and verified according to cleanroom software engineering models.
- Once all the increments are ready, it is integrated as a whole. Thus the functionality of the system grows with the time.
- The task of each increment for cleanroom is explained in the Fig. 2.6.1 below.
- The application or product requirements are developed as discussed earlier i.e. it is started with requirement elicitation then analysis and negotiation, specification, modeling and finally requirement validation and management.
- As soon as the functionality is assigned to software element of the system, the cleanroom increment begins.
- The following sequence of tasks occur :

➢ **Increment planning**

- The incremental plan is developed. This plan adopts the incremental strategy and functionality of each increment in the development schedule is created.
- The projected size, cost and other functionalities is taken into consideration. The only care is taken that each increments are certified and are integrated in proper time according to the schedule or plan developed.

➢ **Requirements gathering**

- Requirement gathering is done using the same technique as described for traditional system development.
- The only difference is that a more detailed description of the customer level requirement is developed for each of the increments.

➢ **Box structure specification**

- In cleanroom strategy, a specification method uses box structures. These box structures describe the functional specification.
- The box structures separate the definition of behaviour, data and procedure in each increment.

➢ **Formal design**

- By using this box structure, the cleanroom design is a natural specification. In this approach, it is possible to make distinction between two activities i.e. specification and component level designs.
- In cleanroom approach, we call specification as black boxes and component level designs as state boxes and clear boxes respectively.

➢ **Correctness verification**

- In cleanroom approach, the cleanroom team conducts rigorous correctness verification activities on the design and then the code.
- Verification begins with black boxes and then state boxes and clear boxes. The black box is the highest level and it is also called as specification level. The next levels are design details and code respectively.

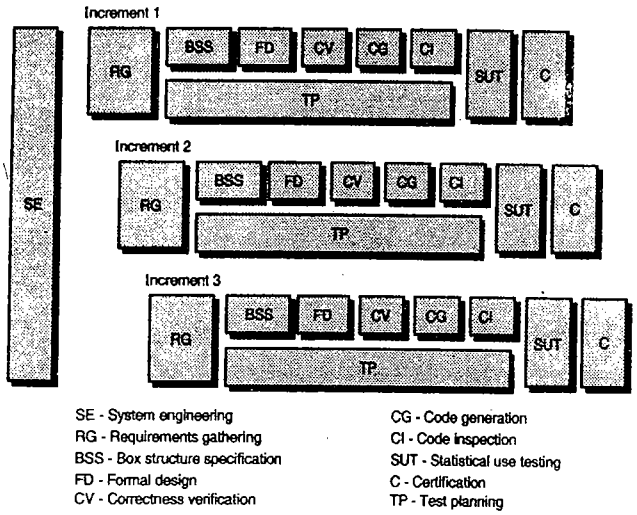


Fig. 2.6.1 : The cleanroom model

➢ **Code generation, inspection and verification**

- The box structure specifications uses specialized language and later they are translated into the appropriate programming language.
- The relevance between code and box structure is verified properly.
- Its correctness is checked properly. The syntactic correctness is also checked. Then overall verification is done for the source code.

➢ **Statistical test planning**

- The projected usage of the software is analyzed and are planned and designed.
- This cleanroom activity is conducted in parallel with specification, verification, and code generation.

➢ **Statistical use testing**

- The traditional testing of computer software is too exhaustive and it is impossible.
- So in cleanroom approach, a finite number of test cases are designed by using statistical use testing.
- In statistical use testing, the series of test cases are derived from statistical samples of entire program executions. These samples are collected from the group of users in some targeted population.

➢ **Certification**

- After completion of verification, inspection and statistical use testing and correction of all errors, the increments are finally certified.
- After this certification the increments are ready for integration in complete system.

- The cleanroom process produces high quality analysis and design models as compared to design models produced by other approaches. The box structure notation is the highlight of cleanroom approach.
- This notation is yet another way of representing requirement and design. The main difference between traditional

approach and cleanroom approach is the verification done on all engineering models.

2.6.3 What Makes Cleanroom Different?

- Differences of the cleanroom approach when the process is defined :
  - o Cleanroom approach uses statistical quality control in software development process with a well defined strategy. This leads to continuous improvement. This improvement is achieved by using cleanroom life cycle.
  - o The cleanroom life cycle focuses on mathematics based software engineering for correct software designs. It also uses statistics-based software testing for certification of software reliability.
- Cleanroom software engineering differs from the traditional software engineering because :
  a) It uses statistical quality control.
  b) It verifies design specification by using a mathematical model for correctness.
  c) It emphasizes statistical use testing for controlling the errors.
- The cleanroom approach uses all the traditional software engineering principles and concepts for software development. In addition it focuses on good analysis and design procedures to achieve high quality.
- The cleanroom approach differs from conventional software practices by eliminating unit testing, debugging and reducing the good amount of testing done by developer. Because cleanroom approach relies on Statistical test planning and Statistical use testing.
- In conventional software development, errors exist even after final delivery of the product. In this approach each module is tested using unit testing to debug errors. But after the delivery, end user finds more and more errors.
- Thus rework is done to debug all these errors and this is time consuming and costly activities. But in cleanroom approach the chances of errors are nearly eliminated as it uses group of population to find errors (Statistical use testing).
- In cleanroom software engineering, we focus on correctness, verification and statistically based testing instead of unit testing and debugging in conventional approach.
- These activities used for continuous improvement, make the cleanroom approach different from conventional approach.

2.6.4 Cleanroom Process Model

SPPU- Dec. 14, May 16, Dec. 16

University Question

Q. Explain the cleanroom process model.
(Dec. 2014, May 2016, Dec. 2016, 10 Marks)

- Cleanroom software engineering uses box structure specification. A "box" consists of the system or some part or some aspect of the system. It uses stepwise refinement process. The boxes make a hierarchy and each box has its own transparency.
- The information in each box is enough to define its refinement without depending on other boxes implementation. It makes developer to separate system hierarchy from low level to top level implementation.
- In cleanroom software engineering, three types of boxes are used :

1. Black box 2. State box 3. Clear box

2.6.4.1 Black Box

- The black box specifies the behaviour of a system.

- It focuses on system specification and functional specification also.
- The system or the part of a system responds to specific events by applying a set of transition rules.

2.6.4.2 State Box

- The state box consists of state data and operations similar to the concept of objects. In state box specification view, the inputs to the state box (some events) and outputs (responses) are represented.
- The state box also represents the history of the black box events. i.e. the data present in the state box that must be retained in all the transitions.

2.6.4.3 Clear Box

- All the transition functions that are used by state box are defined properly in clear box also.

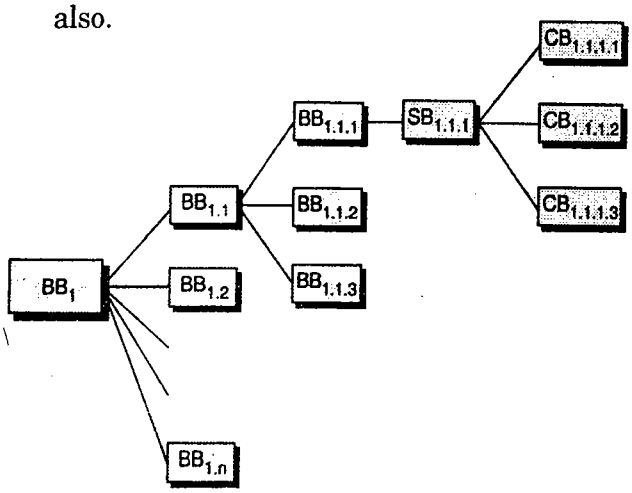


Fig. 2.6.2 : Refinement process

- In short we can say that clear box consists of all the procedural design of previous box i.e. state box.
- Fig. 2.6.2 explains the refinement approach using box structure specification. A Black Box (BB1) shows responses for a complete set of events. BB1 can be further refined into a set of black boxes i.e. BB1.1 to BB1.n. Each of these black boxes shows a class of behaviour.
- The refinement process continues till a cohesive class of behaviour is identified (e.g., BB1.1.1). A state box (SB1.1.1) is then defined for the black box (BB1.1.1). In this case, SB1.1.1 contains all data and services required to implement the behaviour defined by BB1.1.1.
- At last, SB1.1.1 is refined into clear boxes e.g. CB1.1.1.n and also the procedural design details are specified.
- Along with all these refinement steps, the verification of correctness also takes place. The state-box specifications are verified to ensure that each of the boxes conform to the behaviour defined by the parent black-box specification.
- Similarly, clear-box specifications are verified against the parent state box. It should be noted that specification methods based on formal methods can also be used instead of the box structure specification approach. In all these approaches, each level of specification can be formally verified.

2.6.5 Black Box Specification

- In black-box specification, an abstraction, set of events and response are shown in Fig. 2.6.3. The function **f** is applied to the sequence S*, S (input events), and transforms them into an output i.e. response R.
- Most of the concepts used for object-oriented systems are also applicable for the black box approach. Data abstractions and the operations that manipulate those abstractions are encapsulated by the black box. Similar to a class hierarchy, the black box specification also shows the usage hierarchies in which low-level boxes inherit the properties of those boxes higher in the tree structure.

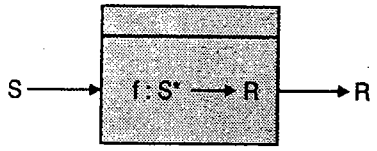


Fig. 2.6.3 : A black-box specification

2.6.6 State Box Specification

- The state box may be considered as a simple state machine. In the transition, a system responds to the events reaches to some new state from the existing state.
- During the transition, an action occurs i.e. the state box determines new state and the action i.e. response will occur as a result of this transition.
- Consider the Fig. 2.6.4 the state box contains a black box. The input event S, to the black box reaches to state T.
- A mathematical description of the function **f** is shown below :

$$g : S^* \times T^* \longrightarrow R \times T$$

where g is a subfunction of state t. The state-subfunction pairs (i.e. t, g) define the black box function **f**. S* is sequence and T* is set of states.

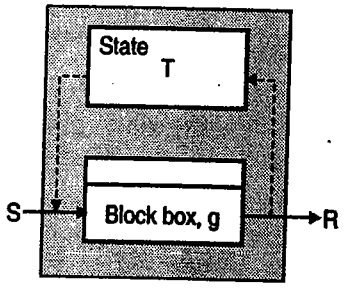


Fig. 2.6.4 : State box specification

2.6.7 Clear Box Specification

- The clear-box specification is very close to procedural design and structured programming.
- In fact, the subfunction g within the state box is replaced by the structured programming that can implement g.
- The clear box shown in Fig. 2.6.5 is considered as the structured programming replacement of black box, g, shown in Fig. 2.6.5. The refinement process takes place as a result.

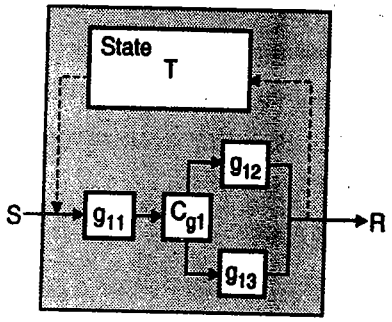


Fig. 2.6.5 : Clear box specification

- Thus the procedural specification shown in the clear-box hierarchy can be proved to be correct.

2.6.8 Cleanroom Design Refinement and Verification

- Each clear-box specification represents the design of a procedure. These procedures or subfunction are important for completing the transition to a new state i.e. state box transition.
- The stepwise refinement with the clear box is done. A program function, f, is refined into a sequence of subfunctions g and h. These in turn are refined into programming constructs (if-then-else and do-while).
- One of the cleanroom team performs formal correctness verification at each level of refinement. To accomplish this refinement process, a set of generic correctness conditions are coupled to the structured programming subfunctions.

2.6.9 Cleanroom Testing

SPPU - May 15, Dec. 15, Dec. 16

University Questions

Q. What is the goal of cleanroom testing ? **(May 2015, Dec. 2015, 5 Marks)**

Q. How do we certify a software component during cleanroom testing ? **(Dec. 2016, 8 Marks)**

- The strategy and tactics of cleanroom testing are basically different from traditional testing strategies and tactics.
- Traditional methods generate a set of test cases to find the design and coding errors.

The primary goal of cleanroom testing is to validate software requirements by showing the statistical samples of use-cases that have been executed successfully.

2.6.10 Statistical Use Testing

SPPU - May 15, Dec. 15

University Question

Q. Discuss in brief the statistical use testing. How do we certify a software components in cleanroom testing ? (May 2015, Dec. 2015, 5 Marks)

- The user of a computer program does not need to understand the technical details of the design.
- The user-visible behaviour of the program is driven by inputs and events that are often produced by the user. But in complex systems, the possible combination of input and events can be extremely broad.
- This combination is also called use cases. These use cases used by statistical use testing will verify the behaviour of the program.
- To accomplish this, cleanroom testing teams or certification teams should determine a usage probability distribution for the software. The black box specification for each increment of the software is analyzed to define a set of inputs or events that cause the software to change its behaviour.
- Based on various interviews with some potential users, the general understanding of the application domain is done and a probability of use is assigned to each input values. Test cases are generated for each input value according to the usage probability distribution.
- Cleanroom software engineering is being used to develop a software increment and also keeps track on various security issues.
- Consider the five input values identified for this increment. Analysis indicates the percent probability distribution of each input. To make selection of test cases easier, these probabilities are mapped into intervals numbered between 1 and 99 :

Program Inputs	Probability	Interval
Mantle /Dismantle (MD)	40%	1 – 39
Region set (RS)	13%	40 – 53
Query (Q)	20%	54 – 73
Test (T)	17%	74 – 90
Panic button	10%	91 – 99

- In order to generate the test cases that are applied to usage distribution, the number from 1 to 99 is considered. These numbers are chosen in random selection. Each of this random number corresponds to an interval on the usage distribution. Each of these random numbers corresponds to the probability of input value occurrences.
- For example, assume the following random number sequences are generated:

15	89	28	22	35	46
81	20	32	69	39	11
38	22	42	84	83	4

- Selecting the appropriate input based on the distribution interval shown in the table, the following use-cases are derived :

  MD – T – MD – MD – MD – RS

  T – MD – MD – MD – Q – MD – MD

  MD – MD – RS – T – T – MD
- The testing teams verify all the use case and their behaviour in correspondence with the system specification. The timing of all the test cases is recorded, in order to find the time intervals between the test cases. This time interval is very useful in determining the Mean-Time-To-Failure (MTTF).
- In conducting the test case, if failure does not occur for the long time then we can trust on the reliability of the software and it is assumed to be very high.

2.6.11 Differences between the Agile Development and Clean Room Approaches

Sr. No.	Agile development approach	Cleanroom approach
1.	An agile process model includes the concept of development along with a set of guidelines necessary for the development process. The conceptual design is necessary for the satisfaction of the customer. The concept is also necessary for the incremental delivery of the product. The concept or the philosophy makes development task simple. The agility team must be highly motivated, updated with latest skill sets and should focus on development simplicity.	Cleanroom software engineering is defined as an engineering approach that builds correctness in software being developed. Unlike the traditional approach of analysis, design, code, test and debug, it offers some different methodology. The main concept behind cleanroom software engineering is removing the dependency on costly processes i.e. defect removal processes by writing the correct code right from first line and verify its correctness before the testing phase starts.
2.	We can say that agile development is an alternative approach to the conventional development in certain projects.	Cleanroom development is not an alternative approach but it should be incorporated in all the conventional development projects.
3.	The development guidelines emphasize on analysis and design activities and continuous communication between developers and customers. An agile team quickly responds to changes. The changes may be in development, changes in the team members, and changes due to new technology. The agility can be applied to any software process. It emphasizes rapid delivery of operational software.	Cleanroom software engineering incorporates the quality approach of writing the code right from the beginning of the system and finally it accumulates into a complete system.
4.	In this development process customer, business people and developers must work together on daily basis.	In this development process customer, business people and developers are not supposed to work together on daily basis. Some formal meetings are enough during requirement gathering and requirement analysis phase.
5.	The messages are conveyed orally i.e. face-to-face. Conversation is essential.	The cleanroom software engineering approach takes the software development process to a next level and messages are not supposed to be sent orally i.e. face-to-face.
6.	In agile development approach, if software fails, then we can face so many hazards that can challenge the human safety also.	In cleanroom software engineering approach, we can guarantee that it will not lead to any hazards like our traditional systems or agile development approaches, in case of failure of software.

❑❑❑

UNIT I

CHAPTER 3

Software Quality Assurance (SQA)

Syllabus

Software Quality Assurance (SQA): Verification and Validation, SQA Plans, Software Quality Frameworks, ISO 9000 Models, CMM Models.

Syllabus Topic : Software Quality Assurance (SQA)

3.1 Software Quality Assurance (SQA)

SPPU - May 12, May 14

University Question

Q. Write short note on: Software quality assurance (SQA). **(May 2012,May 2014, 6 Marks)**

Concepts and Definitions

SQA (Software Quality Assurance) is a planned and systematic method for evaluation of the quality of software product, standards, processes, and procedures.

- It assures that the standards and procedures once established are followed throughout the software development life cycle.
- The evaluation of compliance with these standards and procedures is done by continuous monitoring throughout the development process, product evaluation, and by conducting audits.
- Software development and different control processes can also add quality assurance approval points. The evaluation process for SQA can be conducted by using the applicable standards.

Standards and Procedures

- The task of establishing standards and procedures for software development process is quite critical. It is because they provide the framework that is actually a platform for software evolution. The established standards are nothing but the established criteria by which the software engineering products are compared.
- The procedures are also the established criteria by using it, the development processes and different control processes are compared.
- The standards and procedures actually establish the methods for developing software applications. The role of SQA is to provide assurance of their existence and adequacy.
- The documentation of standards and procedures is also very important since all the SQA activities relay on definitions of project compliance. Following are some important SQA activities :
  - Continuous monitoring
  - Product evaluation and
  - Auditing.

3.1.1 Types of Standards

1. Documentation Standards
2. Design Standards
3. Code Standards

- The **Documentation Standards** established under SQA define the proper content for planning and control. This documentation standard provides the consistency throughout the development life of a project.
- Similarly the **Design Standards** define the proper form and content of the design product.
- These standards provide rule book for and corresponding methods for translating the software requirements specification into the actual software design and for representing it in the design documentation.
- The **Code Standards** specify the programming language in which the source code has to be written. The code standards also specify various constraints that should be put for usage of the language features. They also specify the language structures, style, patterns, rules for data structures, interfaces, and internal code documentation.
- The procedures must be followed in carrying out a development process. All the development processes must have documented procedures.
- Following are the examples of processes for which procedures are needed :
  - Configuration management
  - Non-conformance reports (i.e. any development process not following the SQA standards properly)
  - Corrective action
  - Testing and formal inspections.
- If the software products are developed as per the NASA DID, then the management plan defined the software development control processes. For example, configuration management for product standards.
- All the standards should be documented properly as per the **Standards and Guidelines DID** in the product specification.

3.1.2 Software Quality Assurance Activities

- The product evaluation and the process monitoring are important SQA activities which give the assurance that the software development processes and the control processes written in the management plan are carried out effectively.
- They also ensure that all the procedures and standards are correctly followed. Products are continuously monitored for checking that it is following the standards and processes. The audits are very important technique to conduct product evaluation and process monitoring.
- The intermittent review of plan also ensures the conformance of standards and procedures laid by SQA activities. The SQA approval points are built directly into these processes.
- Product evaluation is an SQA activity that assures standards are being followed.
- In actual practice, the first products monitored by SQA must be the project's standards and procedures.
- SQA ensures that there are clear and achievable standards and these can evaluate the compliance of the software application product with established standards.
- The product evaluation ensures that the software application product is developed by conforming all the applicable standards as illustrated in the Management Plan.

- The process monitoring activity in an SQA ensures that the appropriate steps are carried out during the development process.
- The SQA standards monitor all the processes by comparing the actual steps carried out with those in the documented procedures.
- The SQA also ensures that the Management Plan specifies the methods that should be used by the monitoring activity of SQA process.
- The basis technique for SQA process is the auditing that looks the entire product and all the processes in depth. This is done by comparing them with the established standards and procedures by the SQA processes.
- The audit is an important activity to review the management plan, technical processes and assurance processes to provide the actual status of the software application product.
- The main idea behind an SQA audit is to ensure that the control procedures are properly followed and the desired documentation is properly maintained. It also ensures that the developer's status reports accurately reflect the status of the activity.
- The SQA product is nothing but an audit report to display the findings and recommendations to force that the development process obeys standards and procedures established.

3.1.3 SQA Relationships to Other Assurance Activities

Following are some of the important relationships of SQA process with management and assurance activities :

1. Configuration Management Monitoring

- SQA ensures that the software Configuration Management (CM) activities are conducted in accordance with the CM plans, standards, and procedures.
- The SQA reviews the Configuration Management plans for compliance with software Configuration Management policies and requirements. It provides the monitoring for non-conformance. The SQA audits the Configuration Management functions for implementing the standards and procedures and prepares the status reports of its findings.
- The Configuration Management activities that are monitored and audited by SQA plan include the following parameters :
  - Baseline control
  - Configuration identification
  - Configuration control
  - Configuration status accounting
  - Configuration authentication.
- The SQA process keeps on monitoring and auditing the software library. The SQA makes sure that the baselines are established and maintained consistently for their use in development and control.
- Software configuration identification (SCI) is consistent and accurate with software programs, modules, units and associated software documents.
- The configuration control is maintained to make it compatible with various critical phases of software development like testing, acceptance and delivery.
- The configuration status accounting is conducted accurately including the recording and reporting of data. It reflects the software's configuration identification and any changes made to the configuration identification, and the implementation status of approved changes.

- The software configuration authentication is established by various configuration reviews and audits. The performance required by the software requirements specification and the configuration of the software is properly maintained in the software design documents.
- The software development libraries give proper code, documentation, and concerned data in their various versions from start of the project to the final delivery of it.
- All the approved changes to baseline software are included in the product consistently and any unauthorized changes are not included.

2. Verification and Validation Monitoring

- The SQA activities ensure Verification and Validation (V&V) activities by using following parameters :
  - Formal technical reviews
  - Time to time inspections and
  - Walkthroughs.
- The SQA roles are observed in the above three parameters i.e. formal technical reviews, inspections, and walkthroughs. The SQA verifies that they are conducted properly.
- The SQA also verifies that any action required is assigned properly and the related documents are maintained and updated properly.
- The FTRs (Formal technical reviews) should be carried out at the end of every life cycle phase. By doing this, the problems will be detected and corrected. It also checks that all the requirements are satisfied.
- Examples of formal technical reviews are :
  - PDR (Preliminary Design Review)
  - CDR (Critical Design Review) and
  - TRR (Test Readiness Review).
- A technical review takes into consideration overall structure of the software product being developed. All the desired requirements must be fulfilled. The technical reviews are the integral part of the development process.
- In FTR (formal technical reviews), the actual work completed is compared with standards and procedures already established by the software organization. The main objective of SQA is to ensure that all these standards and management plans are executed properly.
- An inspection or the walkthrough are the detailed testing of a product by using step-by-step process and each line of code are traversed to uncover any possible error. The SQA activities ensure that the entire process is properly completed.
- The inspection process is used to follow the compliance to the standards.

3. Formal Test Monitoring

- The SQA activities ensure the formal software testing such as acceptance testing. It is carried out in accordance with SQA plans and procedures established earlier.
- The SQA activities review the testing documentation to follow the standards and procedures established.
- The documentation review includes following :
  - Test plans,
  - Test specifications,
  - Test procedures and
  - Test reports.

- The SQA plan observes the testing and takes continuous follow-up to uncover the non-conformances and correcting them. After the test monitoring, SQA ensures that the final software product is ready for delivery.
- The main objective of SQA monitoring for software testing is to ensure :
  - The occurrence of any incident during testing are noted and recorded. Usually these incidents are not expected in the test procedures.
  - All the test reports are accurate and complete.
  - The regression testing is conducted to evaluate the non-conformances and they are corrected.
  - The correction of non-conformances takes place happens before the delivery.
  - The software testing checks that the software satisfies all requirement specifications.
  - The quality of the testing is ensured. It is done by evaluating that the project requirements are satisfied.
  - Software Quality Assurance during the Software Acquisition Life Cycle.
  - In addition to all the SQA activities mentioned in the earlier sections, there are few phase-specific SQA activities that must be conducted during the Software Life Cycle.
  - At the end of each development phase, the SQA concurrence is an important element.
- The suggested activities for each of the development phase are listed below.

1. Software Concept and Initiation Phase

- The SQA activities must be involved in both writing and reviewing the Management Plan. It ensures that the processes, procedures, and standards are appropriate and auditable.
- In this phase, the SQA activities also provide the QA section of the Management Plan.

2. Software Requirements Phase

- In this phase of software development, the SQA plan assures all software requirements are completed and testable.
- It also assures all functional, performance and interface requirements.

3. Software Architectural (Preliminary) Design Phase

The SQA activities in the architectural (preliminary) design phase include following :

- It ensures that the approved design standards are properly followed.
- It ensures that all the software requirements are allocated properly to the software components.
- It ensures that testing verification matrix is available it is updated.
- It ensures that the Interface Control Documents are conforming the standards.
- It reviews PDR documentation and ensures that all action items are resolved correctly.
- It ensures that the approved design is placed under configuration management.

4. Software Detailed Design Phase

The SQA activities that are applied during the detailed design phase are as follows :

- It ensures that the approved design standards are followed.
- It ensures that all the allocated modules are present in the detailed design.
- It ensures that the results of design inspections are present in the design.
- It reviews CDR documentation.
- It ensures that all action items are resolved.

5. Software Implementation Phase

The SQA activities that are implemented during the implementation phase consists of following audit :

- The results of coding and design activities are included in the schedule available in the Software Development Plan.
- Checking the status of all deliverable items.
- The configuration management activities.
- The software development library and
- The Non-conformance reports and necessary action taken.

6. Software Integration and Test Phase

The SQA activities in integration and test phase of software development include :

- Ensures readiness for testing of all components.
- Ensures that all tests are run as per test plans and procedures. If any non-conformance is found, then it is resolved.
- Ensures that test reports are completed and in the correct form.
- Once testing is completed, the SQA activities certify it and ensure that it is out for delivery.

7. Software Acceptance and Delivery Phase

The SQA activities applied during the software acceptance and delivery phase are taking care of assurance of the performance of a final configuration audit.

8. Software Sustaining Engineering and Operations Phase

- In this phase, there are mini-development cycles to improve the software product or to correct the software.
- In this development cycle, the SQA activities conduct the appropriate phase-specific activities as mentioned in the above section.

9. Techniques and Tools

- The SQA activities must evaluate its needs for assurance tools in comparison with the available off-the-shelf tools for applicability to the specific project. If it does not match with off-the-shelf (already existing) tools, then it can be developed.
- The audit, automatic code standards analyzers and inspection checklists are some useful tools.

Syllabus Topic : SQA Plans

3.2 SQA Plans

- The SQA plan provides a clear idea of producing the quality product. The SQA plan is the backbone of software quality assurance activities. We can SQA plan as a template for quality assurance activities.
- The IEEE organization has issued a standard for SQA plan. The main purpose of this standard is to control the quality of the product and cover all quality assurance activities. This standard also describes the

scope and purpose of the document and it evaluate that all the documents are covered by the quality assurance activities.

- There are various sections of SQA plan and are as follows :
  - o Documentation section
  - o Standards, practices and conventions section
  - o Review and audit section
  - o Test section, and finally
  - o Management section
- The **documentation section** illustrates each of the work product and it consists of :
  - o Project documents like plan of the project
  - o Various models like ERD and different classes
  - o Technical documents like test plan or specification plan etc.
  - o User manual for helping user in operating the product i.e. help files.
- The documentation describes the set of work products that are necessary to achieve the high quality.
- The **standards, practices and conventions section** describes the all standards list and the list of practices used in development process. These are :
  - o Coding standards
  - o Document standards, and
  - o Review guidelines
- The **review and audit section** provides the reviews and audits to be conducted by the software development team or any stakeholder of the project. This section provides the overview of the approach for all the reviews and audits conducted.
- The **test section** uses all possible testing techniques and testing procedures. It records all the errors uncovered and the necessary corrective actions taken.
- The **management section** uses various tools and methods that support assurance activities and the tasks listed by SQA standards. Following are some important tasks performed by the management section :
  - It uses configuration management procedures to control the change.
  - This section defines contract management approach.
  - It defines the procedures to control the change.
  - It describes the methods to safeguard and maintain the records.
  - It identifies the necessary training for the SQA plan.
  - It also defines the methods to handle risk management activities like :
    - o Risk Identification
    - o Risk assessment
    - o Risk monitoring and
    - o Risk control etc.

Syllabus Topic : Software Quality Frameworks

3.3 Software Quality Frameworks

- The most software developers will agree that high-quality software is an important goal. In the most general sense, software quality is conformance to explicitly stated, functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.
- The definition emphasizes three important points :
  1. Software requirements are the basis for software quality measurement. If requirements are not satisfied then obviously it will yield low quality software.

2. For the development processes, some standards are specified that define the development criteria. These development criteria will guide software engineering process. If the criteria are not followed properly then the quality of the product will be quite low.
3. There are two types of requirements :
   - o Implicit requirements and
   - o Explicit requirements

Even if the explicit requirements are considered, the quality may be degraded if it fails to meet implicit requirements.

- Software quality is a complex mix of factors that will vary across different applications and the customers who request them.
- Software quality factors are identified and the human activities required to achieve them are described as follows :

1. McCall's Quality Factors
2. ISO 9126 Quality Factors

3.3.1 McCall's Quality Factors

- The factors that affect software quality can be categorized in two broad groups :
  - o Factors that can be directly measured (e.g., defects uncovered during testing) and
  - o Factors that can be measured only indirectly (e.g., usability or maintainability)
- The categorization of factors that affect software quality shown in Fig. 3.3.1, focus on three important aspects of a software product :
  1. Its operational characteristics,
  2. Its ability to undergo change and
  3. Its adaptability to new environments

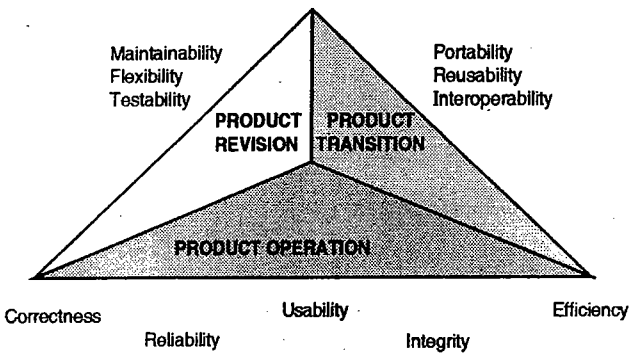


Fig. 3.3.1 : McCall's software quality factors

- Referring to the factors noted in Fig. 3.3.1, provide the following descriptions :

1. **Correctness :** The correctness is the quality factor that assesses the software for its correct functioning according to the specification and customer's requirements.
2. **Reliability :** It is the quality attribute that evaluates the software for its working according to the desired precision.
3. **Usability :** The software usability is defined as the degree to which the application software is easy to use. It defines the efforts taken to operate and learn, submit input and interpret output.
4. **Efficiency :** It is defined as the amount of various resources required to execute a given program. The computing resources are memory devices, processing devices and input output devices etc.
5. **Maintainability :** It is defined as the effort needed to uncover the errors and the problems in functioning of a program and resolve these errors.
6. **Integrity :** The integrity of a system is defined as the efforts taken to manage access authorization. The integrity of a system can be ensured by controlling all the accesses to it.

7. **Flexibility :** The ease with which a program can be modified and extended in future.
8. **Testability :** Amount of time and the efforts taken to test a program.
9. **Portability :** Shifting a program from one environment to another.
10. **Reusability :** How existing codes can be reused in future developments according to the scope of the software.
11. **Interoperability :** Ability to connect one system to the another system.

3.3.2 ISO 9126 Quality Factors

The ISO 9126 standard was developed in an attempt to identify quality attributes for computer software.

The standard identifies six key quality attributes :

1. **Reliability :** The software should be available for the use and should satisfy following key attributes :
   - Maturity
   - Fault tolerance i.e. ability to withstand against failures
   - Recoverability i.e. the system should regain its original state once the failure occurs. It should also ensure the data integrity and consistency while recovering from failure states.
2. **Portability :** Portability means the system should be able to work in different hardware and software environments according to the following attributes :
   - Installability
   - Adaptability
   - Conformance
   - Replaceability
3. **Efficiency :** Efficiency means making the optimal use of the system resources like memory devices, processing devices and input output devices etc. It should take into consideration following attributes :
   - Time behaviour and
   - Resource behaviour.
4. **Maintainability :** Maintainability is defined as the ease with which the software may be repaired and ready to use once again. The maintainability should take into consideration following attributes :
   - Changeability
   - Testability
   - Stability and
   - Analyzability
5. **Functionality :** The degree to which the application software satisfies the customer's needs and requirements according to the following attributes :
   - Accuracy of working model
   - Suitability of the product
   - Interoperability (i.e. the product can be connected to any system),
   - Compliance and
   - Security
6. **Usability :** The software usability is defined as the degree to which the application software is easy to use according to the following usability attributes :
   - Understandability of the software by the end-uses
   - Learnability means leaning the use of the product and
   - Operability.

- The ISO 9126 factors do not support direct measurement of quality, but provides indirect measurement.
- It provides a very good checklist for evaluating the quality.

Syllabus Topic : ISO 9000 Models

3.4 ISO 9000 Models

- The ISO is an organization established to formulate standardization. In this consortium there are 63 countries. ISO (International Standards Organization) published its 9000 series for formulation of standards back in 1987.
- Actually ISO certification works as a reference to compare various software vendors to check the quality. If a vendor does good in its development then can be awarded ISO 9000 certification.
- The ISO 9000 certification is provided if and only if the ISO 9000 series standards are followed. All these standards are designed to improve the quality of the product.
- Following are various variants of ISO 9000 series :
  - ISO 9001
  - ISO 9002
  - ISO 9003
- In the following section, we will focus on some of the limitations of ISO 9000 certification :
  - ISO 9000 does not guarantee 100% for setting up an high quality system.
  - It does not provide any guidelines for defining the process.
  - It is not foolproof.
  - There is no continuous process improvement.

Syllabus Topic : CMM Models

3.5 CMM Models

- The SEI Capability Maturity Model is a process meta-model developed by the Software Engineering Institute (SEI).
- It defines the process characteristics that should exist if an organization wants to establish a software process that is complete.
- The spirit of the SEI Capability Maturity Model should always be adopted. It argues that software development :
  - It must be taken seriously
  - It must be thoroughly.
  - It must be controlled uniformly.
  - It must be tracked accurately.
  - It must be conducted professionally.
  - It must focus on the needs of it's customers
- The SEI Capability Maturity Model presents two types of meta models:
  - As a Continuous model
  - As a Staged model

As a Continuous model

It describes the process in two dimensions as shown in Fig. 3.5.1. Each process area (e.g. project planning, requirement management) are assessed against specific goals and practices and is rated according to following capability levels :

Level 0 : Incomplete

The process area (e.g., requirements management) is either not performed or does not achieve all goals and objectives defined by the SEI Capability Maturity Model for level 1 capability.

Level 1 : Performed

All of the specific goals of the process area (as defined by the SEI Capability Maturity Model) have been satisfied. Work tasks required to produce defined work products are being conducted.

Level 2 : Managed

All level 1 criteria have been satisfied. In addition, all work associated with the process area conforms to an organizationally defined policy; all people doing the work have access to adequate resources to get the job done.

Level 3 : Defined

All level 2 criteria have been achieved. Also the process is tailored from organization's set of standard processes according to organization's guidelines.

Level 4 : Quantitatively managed

All level 3 criteria have been achieved. Also the process is controlled and improved using measurements and quantitative assessment.

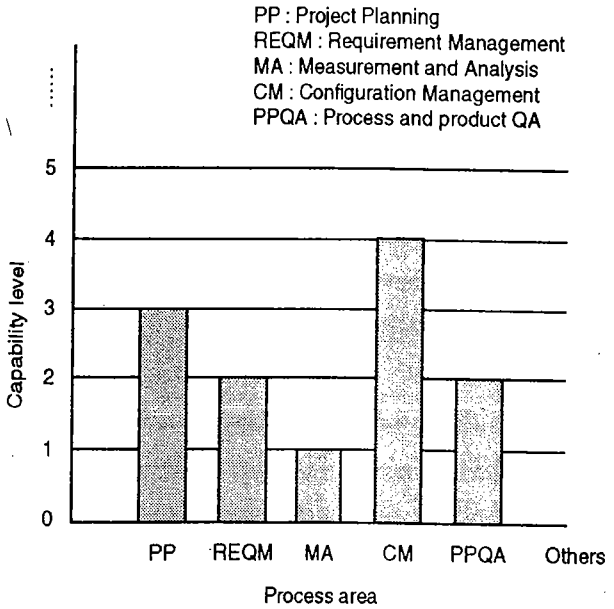


Fig. 3.5.1 : SEI Capability Maturity Model process area capability profile

Level 5 : Optimized

All level 4 criteria have been achieved. Also the process area is adapted all standards to meet changing customer's needs.

The SEI Capability Maturity Model defines each process area in terms of Specific Goals (SG) and Specific Practices (SP).

For example, project planning is one of the process areas. The Specific Goals (SG) and the associated Specific Practices (SP) defined for project planning are :

SG 1 : Establish estimates

SP 1.1 -1 Estimate the scope of the project

SP 1.2-1 Establish estimates of work product and task attributes

SP 1.3-1 Define project life cycle

SP 1.4-1 Determine estimates of effort and cost

SG 2 : Develop a Project Plan

SP 2.1 -1 Establish the budget and schedule

SP 2.2-1 Identify project risks

SP 2.3-1 Plan for data management

SP 2.4-1 Plan for project resources

SP 2.5-1 Plan for needed knowledge and skills

SP 2.6-1 Plan stakeholder involvement

SP 2.7-1 Establish the project plan

SG 3 : Obtain commitment to the plan

SP 3.1 -1 Review plans that affect the project

SP 3.2-1 Reconcile work and resource levels

SP 3.3-1 Obtain plan commitment

The SEI Capability Maturity Model also defines set of five Generic Goals (GG) and related Generic Practices (GP).

For example : Generic Goals (GG) and related Generic Practices (GP) for project planning process area are :

GG 1 : Achieve specific goals

GP 1.1 Perform base practices

GG 2 : Institutionalize a managed process

GP 2.1 Establish an organizational policy

GP 2.2 Plan the process

GP 2.3 Provide resources

GP 2.4 Assign responsibility

GP 2.5 Train people

GP 2.6 Manage configurations

GP 2.7 Identify and involve relevant stakeholders

GP 2.8 Monitor and control the process

GP 2.9 Objectively evaluate adherence

GP 2.10 Review status with higher level management

GG 3 : Institutionalize a defined process

GP 3.1 Establish a defined process

GP 3.2 Collect improvement information

GG 4 : Institutionalize a quantitatively managed process

GP 4.1 Establish quantitative objectives for the process

GP 4.2 Stabilize sub process performance

GG 5 : Institutionalize an optimizing process

GP 5.1 Ensure continuous process improvement

GP 5.2 Correct root causes of problems

The staged SEI Capability Maturity Model defines the same process areas, goals, and practices as the continuous model. The primary difference is that the staged model defines five maturity levels, rather than five capability levels. To achieve a maturity level, the specific goals and practices associated with a set of process areas must be achieved.

3.6 Software Reviews

- The software reviews can be considered as the filter in the software processes. The software reviews are applied throughout the development process at different point of time.
- The main motive behind the software reviews is to uncover the errors that can be corrected later on.
- In the development process, various software activities are implemented i.e. software analysis, software design, software coding etc. The main purpose of software reviews is to purify all these activities.
- The technical reviews are having their own significant since the developers can find errors of others but it is very difficult to find their own errors. The reviews conducted by a person or the group of people give the following results :
  - o The needed or required improvement is highlighted by them.
  - o In some parts improvement is not needed, and
  - o Achieve the technical work more manageable so that quality can be improved.
- There are various types of reviews that can be conducted by the single person or by the team or the group. These reviews may be formal and informal.
- Some of the forms of reviews are as follows :
  - o An informal discussion at the coffee shop is a type of review.
  - o A formal presentation can be delivered for the software design to the group of people like customers, management and technical staff members.
- A formal technical review can also be conducted. It is also called as walkthrough or inspection. Thus it is a good tool to improve the quality of the software.

3.7 Formal Technical Reviews (FTR)

- The formal technical review or FTR is one of the quality assurance activities conducted by the software developers.

- The FTR can be conducted by the other stakeholders of the project also at different point of times.
- Following are some important objectives of the formal technical reviews :
  - Detect the possible errors in the program logic and uncover errors during implementation of the product.
  - Evaluate that the product satisfy the clients' requirements.
  - The product is built according to the standards defined earlier.
  - The product design is uniform, and
  - Ensure that the project is manageable.
- The new developers can be benefitted a lot by FTR, since they come across with different approaches for the software development activities like requirement analysis, design and implementation of the design.
- The FTR is assumed as a separate class of software reviews and consists of :
  - Walkthroughs
  - Inspections, and
  - Reviews by individuals.

3.7.1 Review Meetings

Review Question

Q. Explain Review meetings in brief.

- The **review meetings** are conducted that abide by the following rules and constr aints :
  - In the review meeting, there should be at least three to five people involved for better understanding and adaptation of ideas and reviews.
  - There must be prior preparation for the meeting.
  - The review meeting should not exceed two hours.
- By using these constraints all the stakeholders must focus on the specific parts of the product or the project.
- At the end of the meeting, all the members those who have attended meeting must take unanimous decisions for accepting the product or rejecting the product.
- During the review meeting, the reviewer records all the issues on a paper and summarizes all the issues and present at the end of the meeting to all the attendees of the meeting.
- The review summary has following three important questionnaire :
  - Which parts of the product reviewed,
  - Name the people who reviewed, and finally
  - The conclusion of the meeting.

3.7.2 Review Guidelines

Review Question

Q. What are various software review guidelines ?

Following are some **review guidelines** for better software review meetings :

- During the review meeting, always review the product and do not review the producer who produced it.
- The review leader should ensure the healthy and friendly environment and there should not be any misconduct by any attendee of the review meeting.
- The review leader should immediately call the meeting cancelled if situation goes out of control.
- Always establish the agenda of the meeting and adhere to it. The FTR must be on the proper track and schedule.
- For any issue raised, there should be limit in the debate.
- List the problems and do not try to solve all the problems as it is impossible and people will not be agreed upon.

- Always take the note of all the issues discussed.
- There should be some limit for the number of participants. There should not be heavy crowd in the meeting as objective of the meeting will be failed.
- Develop a checklist for the product to be reviewed.
- Allocate proper resources and proper schedule for the review meetings.
- Finally all early reviews must be reviewed.

3.8 Software Reliability

- Unlike other quality factors, software reliability can be measured directly by using historical data. The software reliability is an important quality factor used by most of the developers.
- The **software reliability** is defined as the probability of failure free program in a specified environment.
- When we discuss failure, one question comes into mind. What is failure?
- Failure can be defined as non conformance to the software requirement i.e. the software application does not produce the desired output as per the requirements and these results in failures.

3.8.1 Measures of Reliability and Availability

Review Question

Q. Explain the following terms :
(i) Reliability (ii) Availability

- If we talk about hardware reliability model then we predict failure due to wear rather than failure due to design defects i.e. physical wear due to effect temperature, corrosion, shock etc.
- But when we discuss software reliability then the failure is related to design defects and the failure can be traced to implementation problems.
- Consider a computer based system in that the measure of reliability is Mean-Time-Between-Failure (MTBF) where

$$MTBF = MTTF + MTTR$$

Where,

$$MTTF = \text{Mean-Time-To-Failure}$$

$$MTTR = \text{Mean-Time-To-Repair}$$

- In addition the software availability is defined as the probability that a program is running as per the requirement at a given point of time.

$$\text{Availability} = [MTTF / (MTTF+MTTR)] \times 100\%$$

- The availability measure is more sensitive to MTTR and it is an indirect measure of maintainability of software.

3.8.2 Software Safety

Review Question

Q. Describe Software Safety.

- Software safety is an activity of Software Quality Assurance (SQA) and it focuses on finding and assessing various hazards. These hazards can have the worst effect on the entire system and can halt the system from working.
- If these hazards are identified in the beginning of the process then it is easy to eliminate and control the hazards.
- Once these hazards are identified at system level then analysis techniques are used to find the probability of occurrence and the safety measures can be taken.
- Even if software reliability and software safety are closely related it is very important to understand the significant differences between them.
- Software reliability uses historical data to find the probability of the occurrence of software failure.
- While software safety is a way in that also failures occurs but these failures lead to accidents that are dangerous to life.

❑❑❑

UNIT II

CHAPTER 4

Requirement Capturing

Syllabus

Requirements Capturing: requirements engineering (elicitation, specification, validation, negotiation, prioritizing requirements (Kano diagram) - real life application case study.

Syllabus Topic : Requirement Engineering

4.1 Introduction

SPPU - May 15

University Question

Q. What is requirements engineering ?

(May 2015, 5 Marks)

- Requirements engineering helps software engineers and software developers to better understand the problem and the nature of the problem they are going to work on. It includes the set of tasks that lead to understanding of :
  - What will be business impact of the software ?
  - What the customer wants exactly ?
  - How end user will interact with the system ?
- Software engineers (sometimes also called as system engineer or analyst) and other project stakeholders (managers, customers and users), all participate in requirements engineering.
- Designing and building the elegant computer software will not satisfy customer's requirements. If requirements analysis is wrong, then design will be wrong.
- Ultimately coding will be wrong. Finally, software expectations will not match with outcomes. Hence requirements engineering should be carried out carefully.

4.2 Requirement Engineering

SPPU - Dec. 12

University Question

Q. Explain in detail requirement engineering task.

(Dec. 2012, 3 Marks)

- The requirements engineering is carried out through the execution of following functions. Some of these requirements engineering functions occur in parallel.
- All these seven functions focus on the customer's needs and care must be taken to satisfy it. All these functions collectively form the strong base for software design and construction. These functions are :

1. Inception	2. Elicitation
3. Elaboration	4. Negotiation
5. Specification	6. Validation
7. Requirement management	

4.2.1 Inception

- Inception is beginning and it is usually said that, 'well beginning is half done'. But it is always problematic for the developer that

what should be the starting point i.e. 'from where to start'.

- The customer and developer meet and they decide the overall scope and nature of the problem statement. The aim is :
  - To have the basic understanding of the problem.
  - To know the people who will use the software.
  - To know exact nature of problem that is expected from customer's side.
  - To maintain effectiveness of preliminary communication.
  - To have collaboration between customer and developer.
- The requirements engineering is a 'communication intensive' activity because a requirement gathering is an initial step for design. Hence exact requirements are gathered with customer communication.
- The developer must ask following questions :
  - Who is behind the request for this work ?
  - Who will use the solution ?
  - What will be the economical benefits of a successful solution ?
  - Is there another source of solution for the same problem ?

4.2.2 Elicitation

SPPU - May 13

University Question

Q. Explain requirement elicitation tasks in brief.

(May 2013, 4 Marks)

- Elicitation means, 'to draw out the truth or reply from anybody'. In relation with requirements engineering, elicitation is a task that helps the customer to define what is required.
- To know the objectives of the system or the project to be developed is a critical job. Why it is difficult to get a clear understanding of customer wants? To answer this question, we have a number of problems like :

Problems of scope

Many times customer states unnecessary technical details. The unnecessary details may confuse developer instead of giving clarity of overall system objectives.

Problems of understanding

Sometimes both customer as well as developer has poor understanding of :

- What is needed ?
- Capabilities and limitations of the computing environment.
- Understanding of problem domain.
- Specify requirements those conflict with other customers and users.

Problems of volatility

Volatility means 'change from one state to another'. The customer's requirements may change time to time. This is also a major problem while deciding fixed set of requirements.

4.2.3 Elaboration

SPPU - May 13

University Question

Q. Explain requirement elaboration tasks in brief.

(May 2013, 4 Marks)

- Elaboration means 'to work out in detail'. The information obtained during inception and elicitation is expanded and modified during elaboration.
- Now requirements engineering activity focuses on developing the technical model of the software that will include :
  - Functions
  - Features
  - Constraints
- Thus, elaboration is an 'analysis modeling' action. This model focuses on 'how the end user will interact with the system'.

4.2.4 Negotiation

SPPU - May 15

University Question

Q. Explain activities and the steps used for negotiating software requirements. **(May 2015, 5 Marks)**

- Here, 'Negotiation' means 'discussion on financial and other commercial issues'.
- In this step customer, user and other stakeholders discuss to decode :
  - To rank the requirements
  - To decide priorities
  - To decide risks
  - To finalize the project cost
  - The impact of above issues on project cost and delivery date.

4.2.5 Specification

- The specification is the final work product produced by requirement engineer. The specification may take different forms :
  - A written document
  - A set of graphical model
  - A formal mathematical model
  - A collection of scenarios
  - A prototype
  - Any combination of above
- The specification is considered as the foundation of all subsequent software engineering activities.
- The specification describes the function and performance of the computer based systems. The specification also describes the constraints are necessary in its development.

4.2.6 Validation

- All previous work completed will be just useless and meaningless if it is not validated against the customers' expectations.
- The requirement validation checklist includes :
  - All requirements are stated clearly ?
  - Are the requirements misinterpreted ?
  - Does the requirements violate any system domain constrains?
  - Is the system requirement traceable to the system model that is created?
  - Are the requirements associated with performance, behaviour and operational characteristics ?

4.2.7 Requirement Management

- Requirement management is a software engineering task that helps the project team members to identify the requirements, control the requirements, track the requirements and changes to requirements at any time as the project exceeds.
- The requirement management task starts with identification and each of the requirements is assigned a unique identifier.
- After the requirements finalization, the traceability table is developed. Traceability table relates the requirements to one or more aspects of the system or it is environment. The traceability tables are used as requirements database and are useful in understanding how change in one particular requirement will affect other parts or aspects of the system.
- Following are some examples of traceability table :
  - **Features traceability table** observes product features and make sure that how it is closely related to customer requirement.
  - **Source traceability table** is used to identify the source of each of the requirement.

- **Dependency traceability table** observes the relationships among requirements.
- **Subsystem traceability table** classifies the requirements as per the subsystem they govern.
- **Interface traceability table** indicates the internal and external system interface relate as per the given requirement.

Requirement \ Specific aspect of the system or its environment	A01	A02	A03	A04	A05				All
R01			✓		✓				
R02	✓		✓						
R03	✓			✓					✓
R04		✓			✓				
R05	✓	✓		✓					✓
Rnn	✓		✓						

Fig. 4.2.1 : Generic traceability table

4.2.8 Initiating the Requirement Engineering Process

- In requirements engineering process, following are considerable issues :
  - Customers may be located at different cities or countries.
  - Customers do not have clear idea of product requirements.
  - Different customers may have different and conflicting opinions about the system to be built.
  - Customers may have limited technical knowledge.
  - Customers may have only limited time to interact with requirement engineer.
- Hence, considering all these issues, following are the steps required to initiate the requirements engineering process :

Identifying the Stakeholders SPPU - May 12

University Question

Q. Explain various stakeholders involved in the project along with their viewpoints. **(May 2012, 6 Marks)**

- Stakeholder is anyone who has direct interest in or benefits from the system that is to be developed. Hence, we can list following people as stakeholders :
  - Business operations manager
  - Product managers
  - Marketing people
  - Internal and external customers
  - End users
  - Consultants
  - Product engineers
  - Software engineers
  - Support and maintenance engineers
  - Others
- All these people have different view of the system regarding the system that is to be developed.

Recognizing Multiple Viewpoints

As many stakeholders exist, they all have different views regarding the system that is to be developed.

For example

- Marketing people are interested in functions and features having potential market.
- Business people are interested in functions and features those can be set within minimum budget.
- End users are interested in functions and features those will be easy to understand and use.
- Software engineers are interested in functions and features those support their existing infrastructure.

- Support engineer may focus on the maintainability of the software.
- It's a duty of software engineer to consider all these viewpoints of all the stakeholders and find the consistent and balanced set of requirements.

Working towards collaboration

Customers must collaborate with themselves and software engineering practitioners to get successful system. The job of requirement engineer is to find :

1. **Common areas :** The requirements for which all stakeholders agree.
2. **Conflict areas :** The requirements that are purposed by one stakeholder but opposed by another stakeholder.

Now, the higher authorities like business manager or senior technologist can take the decision for the requirements either to forward or to reject these requirements.

Asking the First Questions

The requirements engineering is a 'communication intensive' activity because 'requirements gathering' is an initial step for design. Hence following three sets of questions are used to gain to understand the requirements :

1. **First set : The context free questions**
   - Who is behind the request for this work ?
   - Who will use the solution ?
   - What will be the economical benefits of a successful solution ?
   - Is there another source of solution for the same problem ?
2. **Second set : Questions to gain preliminary understanding of problem**
   - How will you characterize good output that will be generated by successful solution ?
   - What are the probable problems for this solution ?
   - What is the business environment in which this solution will be used ?
   - What are special performance issues and constraints that will affect the solution ?
3. **Third set : Questions those focus on effectiveness of the communication activity**
   - Are the questions official ?
   - Are the questions related to the problem ?
   - Are the questions 'too many' in quantity ?
   - Can anyone else provide additional information ?
   - Are some questions left to ask ?

4.3 Requirement Elicitation

- Elicitation is a task that helps the customer to define what is required. 'Eliciting requirements' step is carried out by series of following steps :

1. Collaborative requirements gathering
2. Quality function deployment
3. User scenarios
4. Elicitation work product

4.3.1 Collaborative Requirements Gathering

- Gathering software requirements is team oriented activity. All the software team members, software engineering manager, members of marketing, and product engineering representatives all work together. The aim of this activity is :
  - To identify the problem.
  - To suggest the solution.
  - To negotiate different approaches.
  - To specify the preliminary set of solution requirements.

- The meeting for 'collaborative requirements gathering' is conducted to discuss all above issues.
- The basic guidelines for conducting a collaborative requirements gathering meeting are :
  - Meeting is conducted and attended by both software engineers as well as customers.
  - An agenda is suggested that is formal enough to cover all points those are to be discussed in the meeting.
  - A 'facilitator' may be customer, developer or outsider controls the meeting.
  - A definition mechanism including work sheets, charts, wall stickers, chat rooms, projectors is used.

4.3.2 Quality Function Deployment

- Quality function deployment is a technique that translates the customer's needs into technical requirements for software.
- In other words **'Quality Function Deployment'** defines the requirements in a way that maximizes customer satisfaction. Quality function deployment includes three types of requirements :

1. Normal requirements
2. Expected requirements
3. Exciting requirements

1. Normal requirements

SPPU - May 12, May 14

University Question

Q. What is meant by normal requirements.

(May 2012, May 2014, 3 Marks)

These are the requirements clearly stated by the customer. Hence these requirements must be present for customer's satisfaction.

For example

- Graphic displays.
- Specific system functions.
- Specified output formats.

2. Expected requirements

These requirements are implicit type of requirements. These requirements are not clearly stated by the customer but even then the customer expects them.

For example

- The developed system must provide easy human machine interaction.
- The system should be menu driven,
- All hot key buttons help should be provided.
- The system should be 'user friendly'.
- The system should be easy to install.

3. Exciting requirements

SPPU - May 12, May 14

University Question

Q. What is meant by exciting requirements.

(May 2012, May 2014, 3 Marks)

These requirements are neither stated by the customer nor expected. But to make the customer more satisfied, the developer may include some unexpected requirements. For example, in word processing software development, only standard capabilities are expected. But, it will be a surprise for the customer if 'page layout capabilities' and 'advanced graphical features' are added.

4.3.3 Usage Scenarios

- It is just impossible to move into more technical software engineering activities until the software team understands how these functions and features will be used by different classes and end users.
- To understand this, the developer and users create a set of scenarios those will identify all these issues. The scenario is called as **'Use Cases'**. Details of Use Case diagrams are included in next chapter.

4.3.4 Elicitation Work Product

- The work products produced by requirement elicitation depend upon the size of the system or the system to be built.
- The information produced as a consequence of requirements gathering includes :
  - A statement of need and feasibility.
  - A statement of scope for the system or product.
  - A list of customers, users and other stakeholders who participated in requirement elicitation.
  - Description of system's technical environment.
  - The list of requirements and domain constraints.
  - The set of scenarios.
  - Any prototype developed to define requirements clearly.

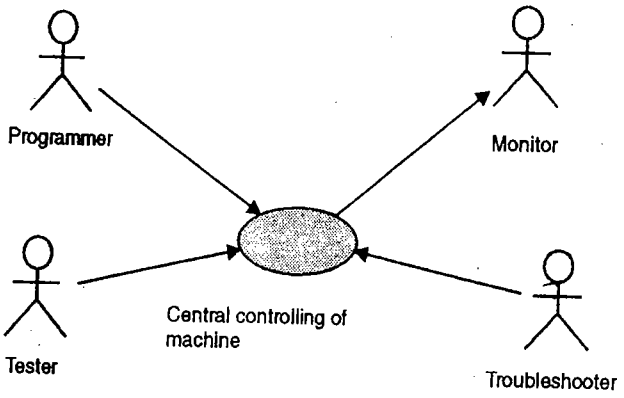
4.3.5 Elicitation Techniques

- These are the data collection techniques.
- They are used in cognitive science, knowledge engineering, linguistic management, psychology, etc.
- Knowledge is directly acquired through human being.
- It includes various techniques including interview, observations, participatory design, focus groups, etc.

4.3.6 Developing Use Cases

- A Use case exhibits the behaviour of the system according to the response received from any of the stakeholders. Alternatively a use case explains how the users will operate the system under any specific circumstances.
- Use cases are defined from actor's point of view. An actor is a role that refers the user or device that interacts with the software.
- The Use case diagram shows the relationship between actors (e.g. person, machine, another system etc) and use cases (sequence of actions i.e. procedures /functions).
- Note that the actor and end user are not necessarily the same thing. A typical user may play number of different roles when using the system, whereas an actor represents a class of external entities that plays just one role in the context of use case.
- **For example**

  Consider the application where the machine operator handles control computer for manufacturing cell. Here, machine operator is a user.



Actors and use case for controlling the machine in manufacturing cell of factory

Fig. 4.3.1

- The software for control machine requires four different modes for interaction : Programming mode, test mode, monitoring mode, troubleshooting mode.
- Hence four actors can be defined : programmer, tester, monitor and trouble shooter.
- **There are two types of actors**

  1. Primary actors
  2. Secondary actors

1. Primary actors

These actors interact with the system to achieve required system function and

derive intended benefits from the system. They work directly with the software.

2. Secondary actors

- These actors support the system so that primary actors can do their work. After identifying the actors the use cases can be developed. Jacobson suggests the number of questions those are answered by the use cases.
- What are primary and secondary actors ?
- What are the goals of actors (i.e. primary and secondary actors) ?
- What are the preconditions that exist before the development begins ?
- What are tasks and functions that are performed by actors ?
- What are considerable exceptions ?
- What are the possible variations in primary and secondary actor's interaction ?
- What are the system information that a actor can acquire, produce ?
- What are the system information that a actor can modify ?
- Is it necessary for a actor to inform the system, the possible changes in the external environment ?
- What is the desired information of a system that an actor want to know ?
- Is it necessary for a system to inform the actor about unexpected changes ?

4.4 Requirement Specification

- Throughout the analysis model, software engineer focuses **'what'** type of queries.
- **For example**
  - What is the information processed by the system ?
  - What functions the system performs ?
  - What is the behaviour of the system ?
  - What interfaces the system defines ?
- The **'analysis model'** acts as a bridge between **'system description'** and the 'design model'. After considering the requirements at system level in 'system description' the analysis model focuses on the system requirements.
- The information, functions and behaviour of the system defined by 'analysis model' are translated into architectural, interface and component level designs in 'design modeling'.

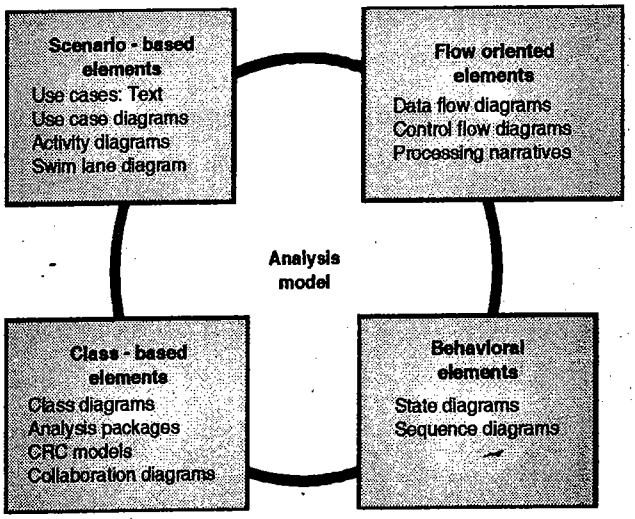


Fig. 4.4.1 : Elements of analysis model

Elements of Analysis Model

The elements of analysis model include :

1. Scenario-based elements
2. Flow oriented elements
3. Class based Elements
4. Behavioural elements

1. Scenario-based elements

They represent the system in user's point of view. Scenario-based elements are :

- Use case - Text
- Use case diagrams
- Activity diagrams
- Swim lane diagrams

2. Flow oriented elements

They provide necessary information that how data objects are transformed by processing the functions. Flow oriented elements are :

- Data flow diagrams
- Control Flow diagrams
- Processing narratives

3. Class based elements

They define the objects, attributes and relationships. Class based Elements are :

- Class diagrams
- Analysis diagrams
- CRC models
- Collaboration diagrams

4. Behavioural elements

They show the states and it is classes and impact of the events on these states. Class based elements are :

- State diagrams
- Sequence diagrams

4.4.1 Requirement Monitoring

- All the requirements should be monitored properly.
- Proper document should be created during collecting requirement.
- Developers need to check the requirement frequently and accordingly implement it.
- There are many chances that requirement changes frequently.
- We need to monitor all the requirement.

4.5 Requirement Negotiation

- In traditional development process, the requirement engineering has the following phases :
  - Inception
  - Elicitation
  - Elaboration
- These phases are enough to determine the customer's requirement. But, practically it rarely happens that all the requirements are met as per customer's need. In actual practice, one has to go for negotiation with one or more stakeholders.
- The importance of negotiation is to develop a project plan that meets customer's needs, also keeping in mind the other factors like time, people, budget etc.
- The best negotiation yields "win-win" result.
- Boehm has defined a set of negotiation activities at the beginning of each software process iteration :
  - Identification of the system or subsystem's key stakeholders.
  - Determination of the stakeholders' "wins conditions".
  - Negotiation of stakeholders' "wins conditions".
- After successful completion of these initial steps, we achieve a win-win result. Thus we can proceed to next software engineering activities.

4.6 Requirement Validation

SPPU – May 12, May 14

University Question

Q. How requirements are validated?

(May 2012, May 2014, 3 Marks)

- Once requirement model is built, it is checked for inconsistencies and ambiguities. Then the requirements are optimized.
- During the review of the requirements model, following questions arise :
  - Are all the requirements consistent ?
  - Whether requirements followed proper level of abstraction ?
  - Is the requirement really necessary ?
  - Are there any requirements which conflicts with one another ?

- Is each requirement testable, once implemented ?
- Does the requirement model properly reflect the information, function and behaviour of the system to be built ?
- Does the requirements model use the requirements pattern to simplify the model ?

- These questions are answered to make sure that the requirements model is as per the customer's exact needs.

4.7 Prioritizing Requirements

- It is used in software product development.
- It is used to determine which requirement should be included in certain release of the product.
- It is also useful to minimise the risk during product development.
- Several methods are available to prioritize the requirement.

4.7.1 Cost Value Approach

- Cost value approach is one of the approach in prioritizing the requirement.
- It was invented by Joachim Karlsson and Kevin Ryan.
- Priority of the requirement can be decided by following steps.
- Requirement engineers carefully review candidate requirement for completeness of the product.
- Customers and users do the pairwise comparison to access relative values.
- Experienced software engineer uses the pairwise comparison to estimate relative cost of implementation.
- Software engineer uses the relative values to calculate each candidate requirement value and implementation cost.
- Stakeholders use the cost value diagram as conceptual map for analysing and discussing candidate requirement.
- Release planning process consists of sub processes which are as follows.
  - Prioritize requirements
  - Select requirement
  - Define release requirement
  - Validate release requirement
  - Prepare lanch.
- Other prioritization techniques are as follows ;
  - Quality function deployment
  - Binary search tree
  - Planning game
  - 100 point method
  - Software engineering risk
  - Value oriented prioritization method
  - Minimal spanning tree
  - Bubble sort
  - Numeral assignment

4.7.2 Prioritizing Requirements with Kano Model SPPU –Dec. 15, Dec. 16

University Questions

Q. How to prioritize software requirements based on Kano Analysis? **(Dec. 2015, 5 Marks)**

Q. Explain Requirements Prioritization with the help of KANO diagram. **(Dec. 2016, 5 Marks)**

- This analysis allows us to prioritize the requirement as function of customer satisfaction.
- Kano defined four different methods for prioritizing the requirement.
- Different methods are as follows.
  - **Surprise and delight :** This capability differentiates the product from other product. More facility should be provided than that of competitor product.
  - **More is better :** More functionality should be provided.
  - **Must be:** functional barrier to entry without which user will not use the product.
  - **Better not be :** It represents things which dissatisfy customers.
- Kano diagram for prioritizing the requirement is as follows.

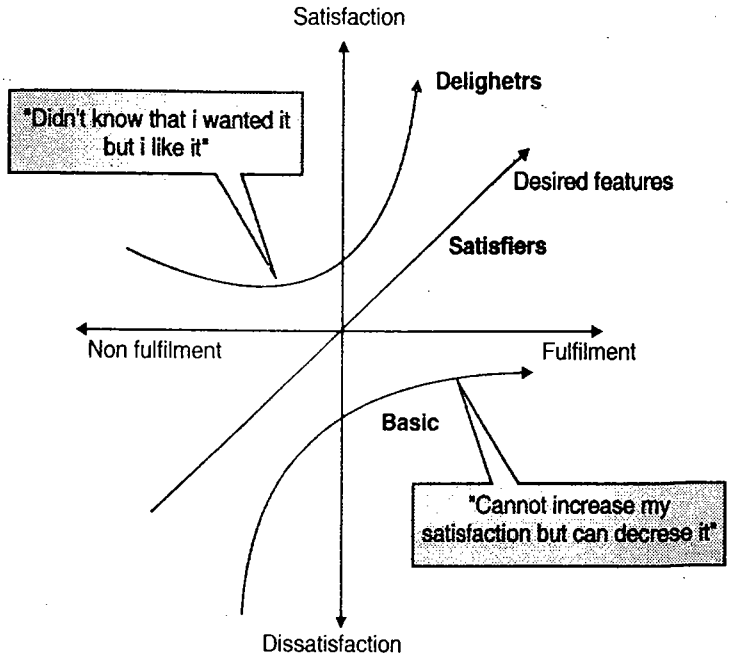


Fig. 4.7.1 : Kano model

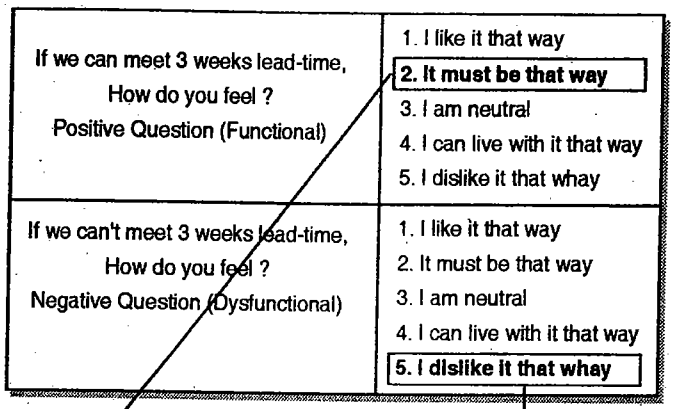
4.8 Real Life Application Case Study

- The most widely used methods to capture the customer's requirement is a survey questionnaire because it can be done at scale.

	5 Points Scale (Linkert)					Rank Order Scaling	
	1 Very Unimportant	2 Very Unimportant	3 Very Unimportant	4 Very Unimportant	5 Very Unimportant		1 = Most Important 5 = Least Important
Requirement 1		X				Requirement 1	5
Requirement 2			X			Requirement 2	3
Requirement 3				X		Requirement 3	4
Requirement 4					X	Requirement 4	2
Requirement 5	X					Requirement 5	1

- The above example illustrates common survey instrument. Linkert scale which consists of 5 or 7 points ranging from the lowest degree to the highest degree literally appears on almost every survey questionnaire. It's also used in conjunction with Rank Order Scaling which asks the customer to rank each requirement so the company can identify high priority requirements.
- Dr. Noriaki Kano (Kano Model) indicates that each requirement is not equally important to the eye of customers.
- Kano Model is not only the conceptual model, but also survey instrument. Here's how to use Kano Analysis:
  - Gather customer's requirement
  - Ask two questions for each customer's requirement
  - Develop Kano Questionnaire
  - Create Kano Evaluation Table

- Suppose a customer says he feels that 3 weeks lead-time is a must (answer 2 in positive question) and he dislikes it if 3 weeks lead-time can't be met (answer 5 in negative question),



Customer requirement		Negative Question (Dysfunctional)					Legend
		1. like	2. must be	3. Neutral	4. live with	5. dislike	M = Must be
Positive Question (Functional)	1. like	Q	A	A	A	O	O = One Dimensional
	2. must be	R	I	I	I	M	A Attractive
	3. neutral	R	I	I	I	M	I = Indifferent (No Preference)
	4. live with	R	I	I	I	M	R = Reverse (can be either way)
	5.dislike	R	R	R	R	Q	Q = Questionable (Wrong answer)

- Since you have to ask many customers, you will need to tally the results to determine how majority of customers expresses their requirements.

	M	O	A	I	R	Q	Category
Requirement 1	54	35	8	3	0	0	M
Requirement 2	20	63	2	14	1	0	sO
Requirement 3	25	20	48	7	0	0	A
Requirement 4	13	55	20	10	1	1	O

- Kano Model was originally developed for product design. Anyway, we have tested it in a supply chain environment and found that this tool is actually pretty good.
- The case study is supply chain improvement plan at the animal feed distributor. They would like to know how they should improve operations in order to be more competitive in the market.

❑❑❑

UNIT II

CHAPTER 5

Requirement Analysis

Syllabus

Requirements Analysis: basics, scenario based modeling, UML models: use case diagram and class diagram, data modeling, data and control flow model, behavioral modeling using state diagrams - real life application case study, software Requirement Specification.

Syllabus Topic : Requirement Analysis

5.1 Requirement Analysis : Basics

- Analysis model uses a combination of text and diagrammatic forms to depict requirements of data, functions and behaviour. So that it will be easy to understand overall requirements of the software to be built.
- The **'Software Engineer'** also called as **'Analyst'** builds the model using requirements stated by the customer.
- Analysis model validates the software requirements and represent the requirements in multiple dimensions.
- Basic aim of analysis modelling is to create the model that represents the information, functions and behaviour of the system to be built.
- These information, functions and behaviour of the system are translated into architectural, interface and component level designs in design modeling.
- Information, functions and behaviour of the system are represented using number of different diagrammatic formats.

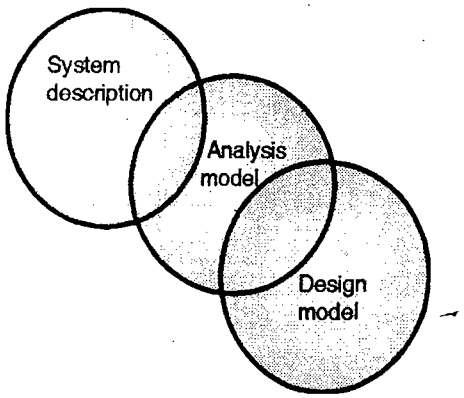


Fig. 5.1.1 : Analysis model, bridge in system description and design model

- As stated previously, the 'analysis model' acts as a bride between 'system description' and the 'design model'.
- The system description describes overall system functionality of the system including software, hardware, databases, human interfaces and other system elements. And the software design mainly focuses on application architectural, user interface and component level designs.
- Thus, all elements of analysis model are directly traceable to the parts of design model. Hence, the analysis model must have following three objectives :
  - To state clearly what customer wants exactly.

- To establish the basis of the 'design model'. The information, functions and behaviour of the system defined by 'analysis model' are translated into architectural, interface and component level designs in 'design modeling'.
- To bridge the gap between a system level description and software design.

5.1.1 Analysis Rules of Thumb

SPPU - May 12, May 13

University Questions

Q. List all the rules of thumb. **(May 2012, 3 Marks)**

Q. What are the rules of thumb? **(May 2013, 3 Marks)**

These rules are

- The requirement analysis must state the requirements at business domain and at high level of abstraction. No need to state the details.
- Each element of analysis model must help for clear understanding of software requirements and must focus on information, functions and behaviour of the system
- Consideration of infrastructure and non-functional model should be delayed to design. For example : Define the databases if required for software under consideration. But no need to consider the details likes classes, functions and behaviour of the databases.
- Try to minimize the coupling throughout the system. The interconnections among different modules is called 'coupling'. That is the coupling measures the 'functional dependency' of different modules. Hence for good design, the interconnection among different components should be minimum.
- The analysis model provides value to all people concern to it.
- **For example :** Customer will use analysis model to validate his requirements. The Designer will use the analysis model for designing the 'design model'. End user can use same model for testing.
- The analysis model must be as simple as possible.
- Only simple model will help the easy understanding of software requirements. Hence analysis model should be simple enough.

5.1.2 Domain Analysis

SPPU - Dec. 12, May 14

University Question

Q. Explain domain analysis. **(Dec. 2012, May 2014, 3 Marks)**

- By definition, software domain analysis is "the identification, analysis and specification of common requirements from a specific application domain." These common software domains can be reused for multiple projects within that application domain.
- In short, software domain analysis leads to 'reusable software components' in software development. Here, in specific application domain common objects, common classes, common frameworks can be identified and can be reused.
- Role of 'domain analyst' is same as job of toolsmith. The job of toolsmith is to design the tools those are used by many people for similar works. The role of domain analyst is to define reusable objects, classes, frameworks those can be used by many people in similar applications.
- **For example :** The specific application domain' may be 'bus reservation system'. The software domains of 'Us reservation system' can be used for 'railway reservation system'.

Advantages

- Use of such software domain analysis saves the time.
- It also reduces the development cost.

5.1.3 Requirements Modeling Approaches

Following are different requirement modeling approaches :

- **Structured analysis :** It consists of **data** and the **process** that transforms data.
- **Object-oriented analysis :** It emphasizes on the classes that collaborates with one another to focus on customer's requirements.

5.2 Scenario Based Modeling : UML Models

SPPU - May 15

University Question

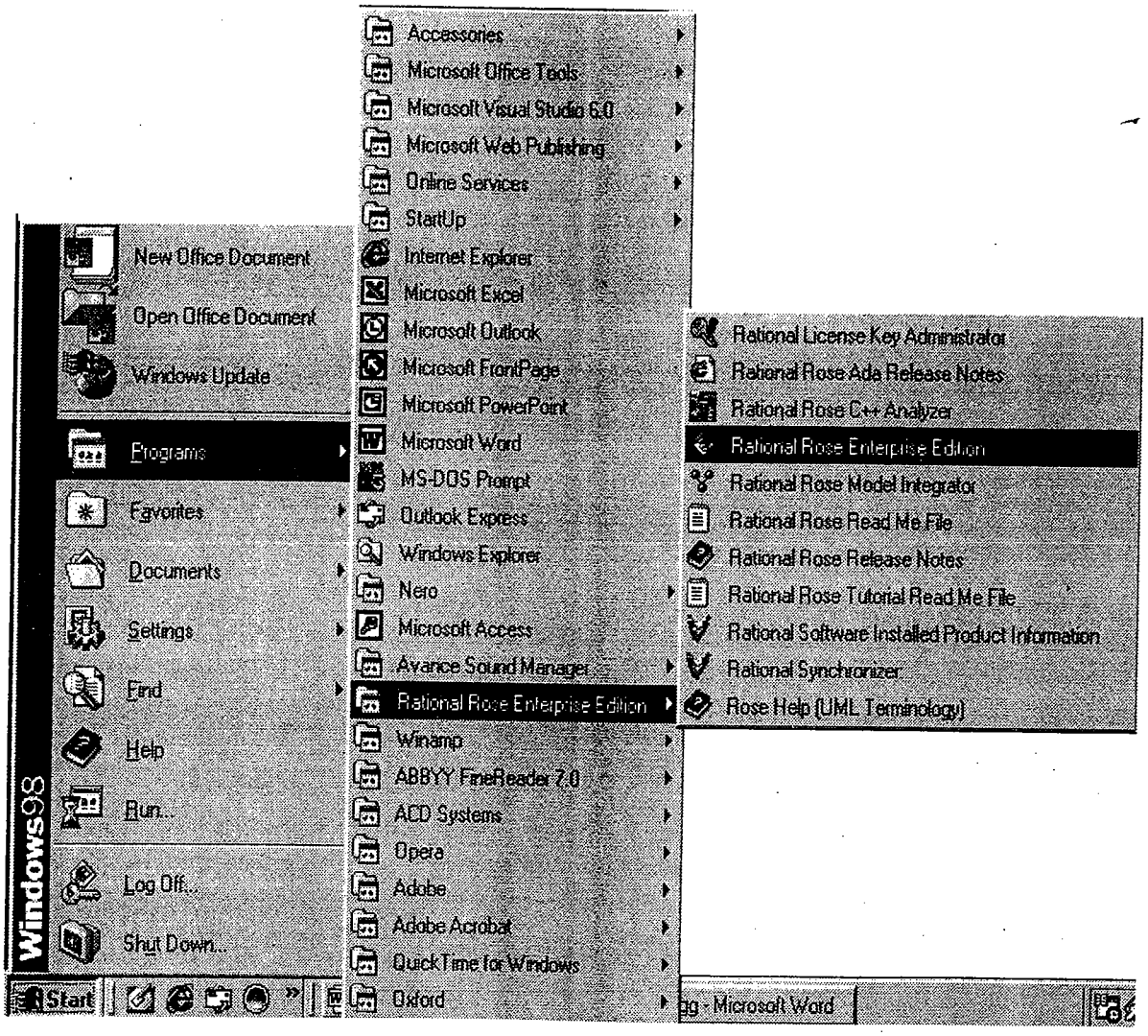
Q. Describe the steps of scenario based modeling with a suitable example. **(May 2015, 5 Marks)**

- Analysis modeling with UML begins with the creation of scenarios.
- In Scenario Base modeling the system is represented in user's point of view. Scenario-based elements are :
  1. Use case diagrams
  2. Activity diagrams
  3. Swim lane diagrams

5.2.1 Diagramming in UML

During 1990 Rumbaugh, Booch and Jacobson combinely presented a Unified Modeling Language (UML) that includes notations for modeling and development of OO systems. By 1997 UML became a industry standard for Object Oriented software development and Rational corporation developed automated tools to support UML methods (i.e. Rational Rose software).

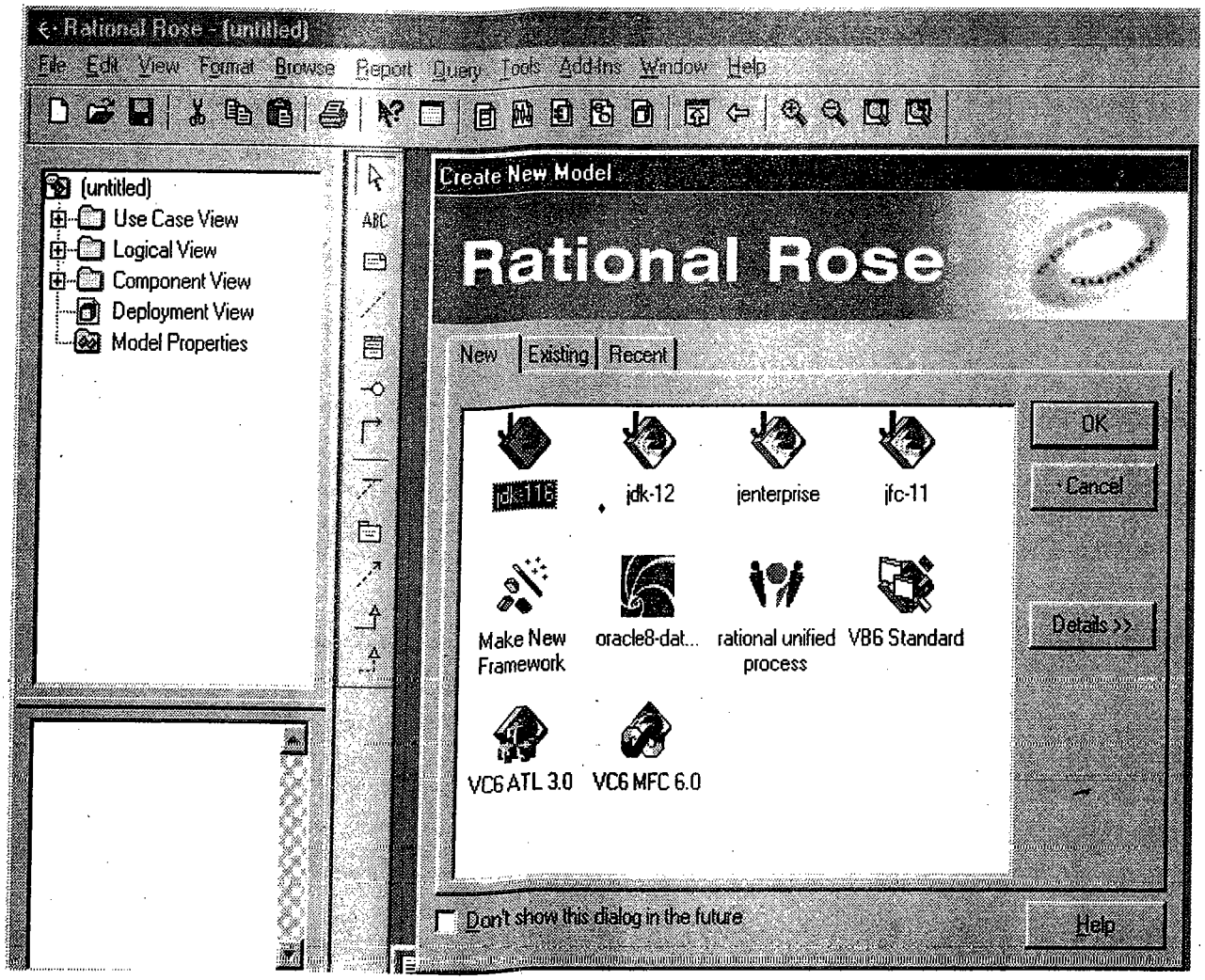
Using Rational Rose for UML diagrams

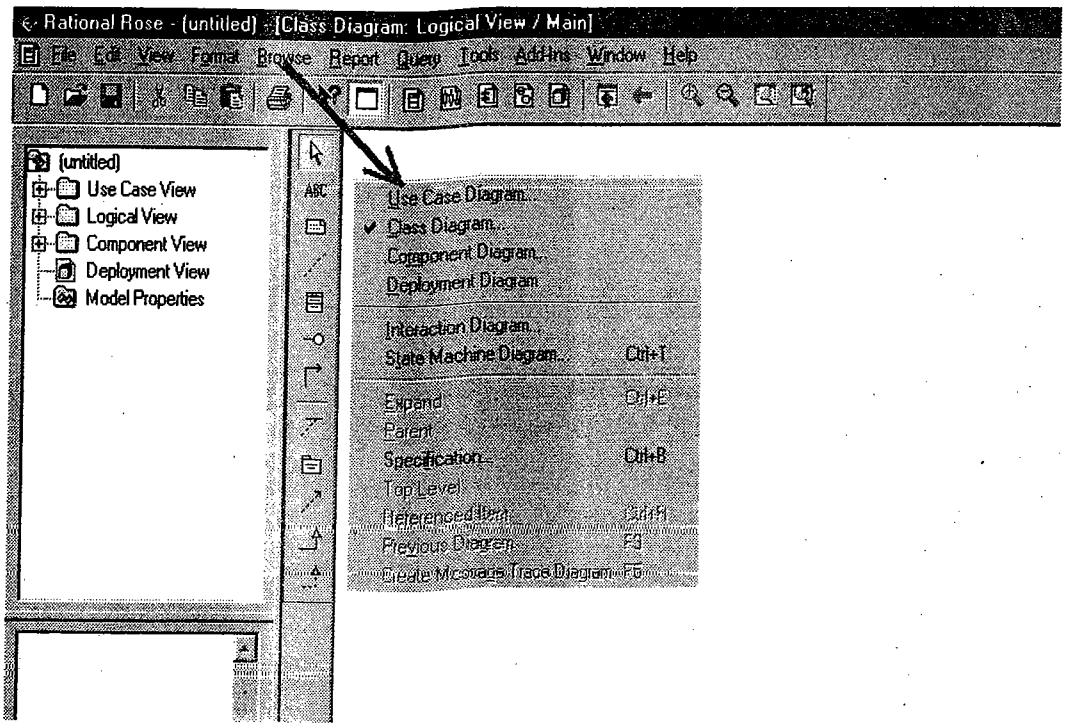


- Install Rational Rose on your PC.

  Go to start menu and open "Rational Rose Enterprise Edition".

- Now select the diagram under "Browse " option to draw specific diagram :





5.2.2 Developing Use Cases Diagram

SPPU - May 12, May 13

University Question

Q. Explain in detail UML diagram stating purpose and applicability : Use- Case diagram.

(May 2012, May 2013, 3 Marks)

- To start developing a set of use-cases, the functions or the activities performed by actor are listed. The list of Use-cases may be obtained from one of the following sources :
  - o From customer communication.
  - o End user communication.
  - o By evaluating the activity diagrams.
  - o Different notations used for use case diagrams are :

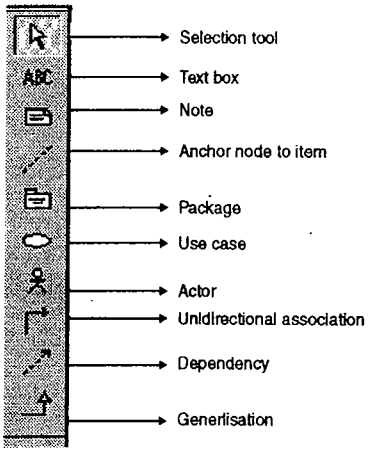


Fig. 5.2.1 : Notations in use case diagram

Example of use case diagram

Problem statement : To develop the Use case diagram for 'food ordering system' in hotel.

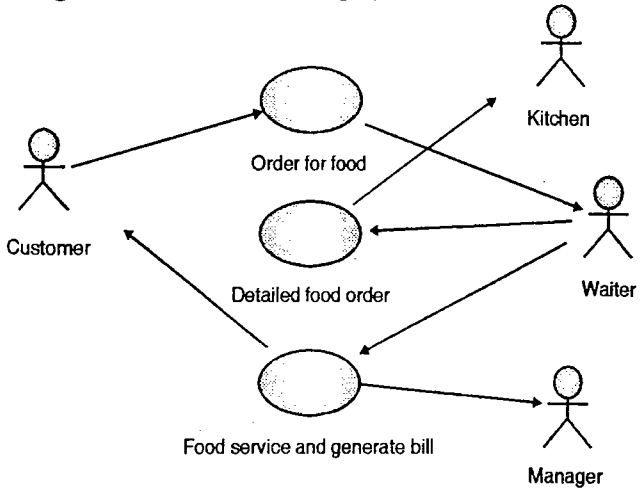


Fig. 5.2.2 : Use case diagram for food ordering system

Explanation

Customer orders for the food. The waiter receives this order. The waiter then forms detailed order. The detailed order is submitted to the kitchen and desired service is provided to the customer. Finally bill is produced. The bill is submitted to customer and bill report is sent to manager.

5.2.3 Developing Activity Diagram

SPPU - May 12

University Question

Q. Explain in detail UML diagrams stating purpose and applicability: Activity diagram.

(May 2012, 3 Marks)

Activity diagrams are same similar to that of flowcharts. The notations used in activity diagram are :

- Rounded rectangles imply specific system functions.
- Arrows represent flow through the system.
- Decision diamonds are used to depict branching decisions.

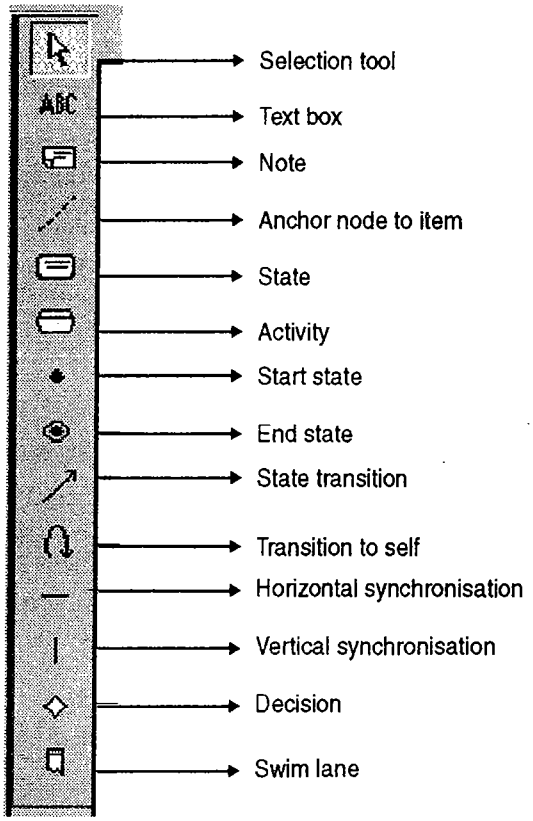


Fig. 5.2.3 : Notations in activity diagram

Example of activity diagram

Problem statement : To develop the activity diagram for 'Building Plan'.

Explanation

- First 'dig foundation' activity will be completed. After completing foundation, two activities 'build walls' and 'connect service' are two activities those can run parallel.
- After completing these activities 'build roof' activity can be started.
- While 'wiring' and 'carpentry' is going on sequentially, at the same time 'fit windows' activity can be completed parallel. Finally, 'painting' activity is completed.

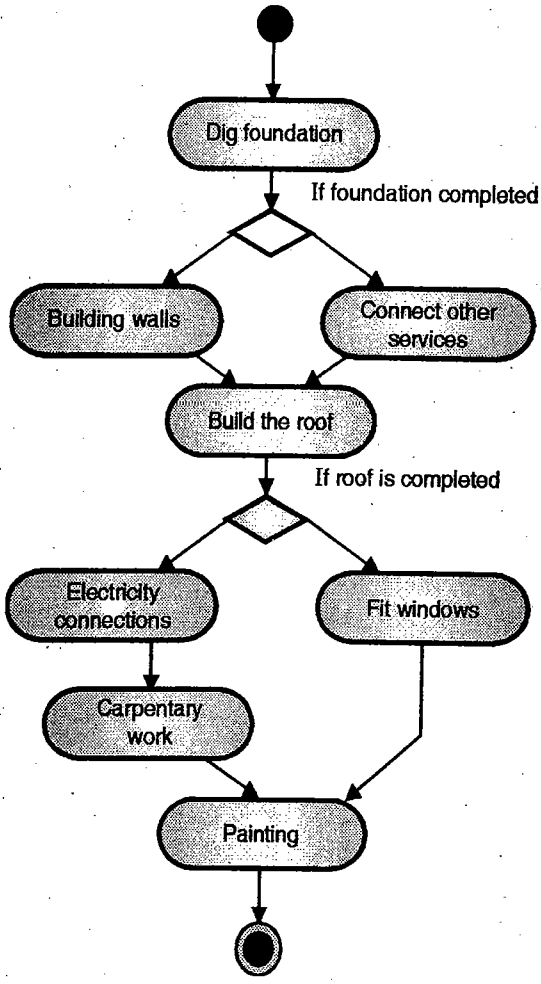


Fig. 5.2.4

5.2.4 Swim Lane Diagram

- Swim lane diagrams are the variation in activity diagrams. Swim lane diagrams allows to represent, the flow of activities described by use cases.
- At the same time these diagrams indicate which actors are involved in specific functions.
- We can also show the interaction between these different activities shown in different lanes.
- To show such parallel activities, diagram is divided vertically into lanes, similar to lanes of swimming pool. Hence these diagrams are called as 'swim lane' diagram.

Example of swim lane diagram

Problem statement

To develop the swim lane diagram for 'purchase equipment order'.

Explanation

- First, Head of the department will submit the requirement for purchasing the equipments.

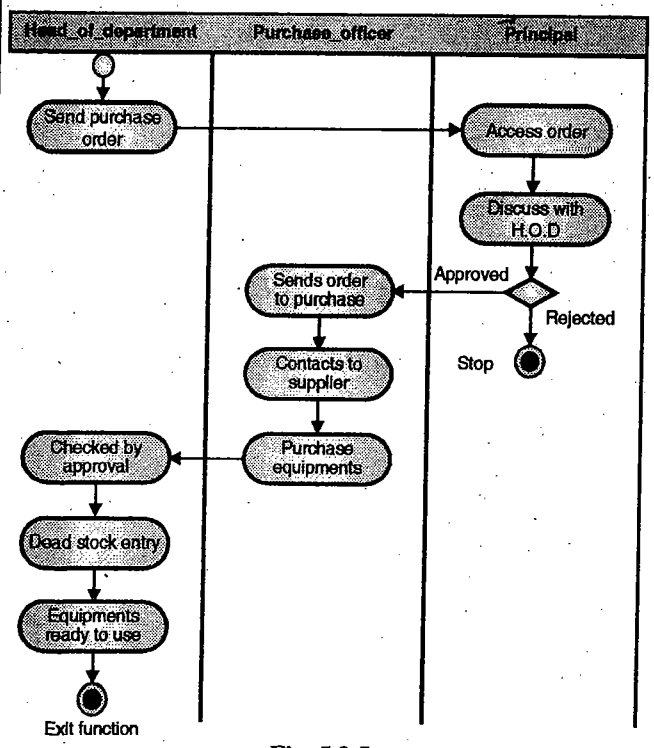


Fig. 5.2.5

- The order is reviewed by the principal.
- If the principal permits the proposal, the order is forwarded to purchase officer.

- Purchase officer now contacts to supplier.
- After purchasing the required equipments, they are checked by H.O.D.
- Finally, the dead stock entry is done in dead stock of the department.
- Purchased equipments are ready for use.

5.2.5 Class Diagram

Example class diagram

The class diagram for computer based system is shown with two important symbols :

Aggregation

Aggregation denotes super class to subclasses relationship.

For example in Fig. 5.2.7, we can say that Monitor, Printer and Plotter are subclasses of super class 'Output Device'.

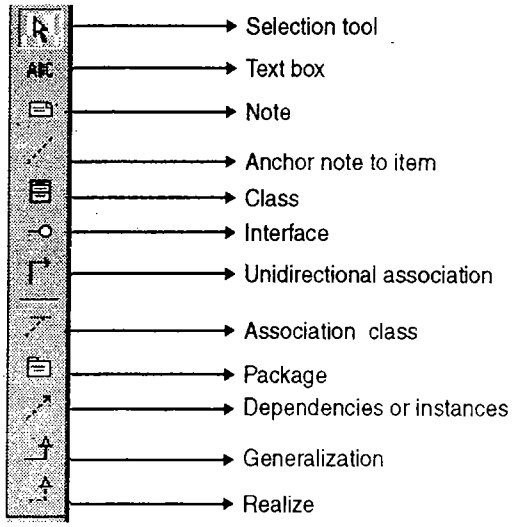


Fig. 5.2.6 : Notations in class Diagram

Generalization

This symbol is used to denote 'is part of' or 'is composed of' relation.

For example : In Fig. 5.2.7 we can say that class CPU is composed of classes RAM, Control Unit and ALU.

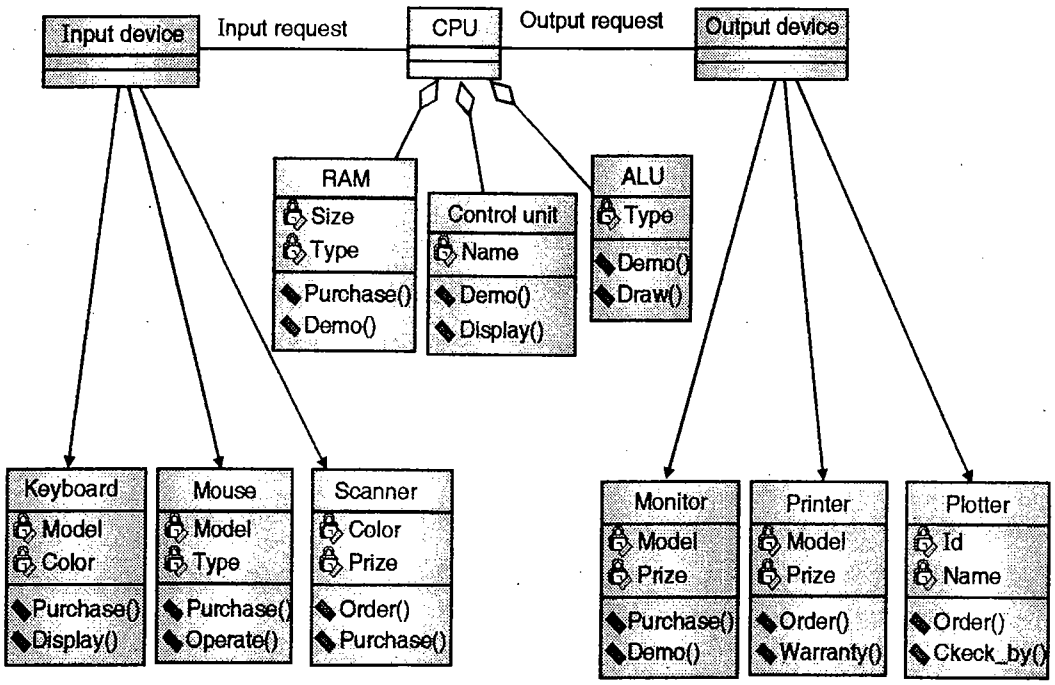


Fig. 5.2.7

Associations and Dependency

Associations

Two classes may be related to one another. This relation is called as 'association'. For example : In computer based system shown in Fig. 5.2.7, class 'input device' is associated with class 'CPU' and class 'CPU' is associated with class 'output device'.

Multiplicity

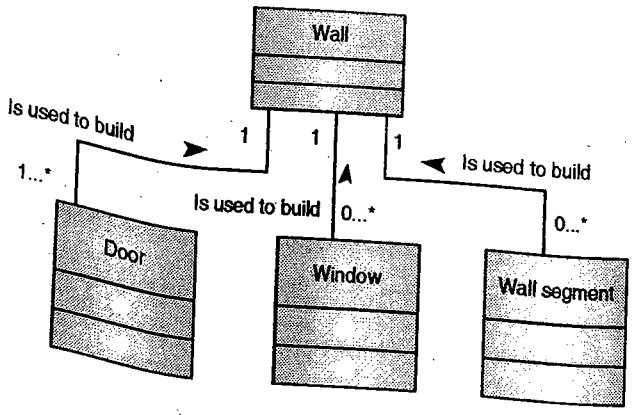


Fig. 5.2.8

The association is further defined by indicating multiplicity. It shows how many occurrences of one class are related to how many occurrences of another class. For example, from Fig. 5.2.8, we can read that 1...* doors are used in building one wall.

Dependency

Sometimes the application requires showing the dependencies between the classes.

For example : The librarian will get access to the 'server system' if he enters the correct 'password'. This dependency is shown in Fig. 5.2.9.

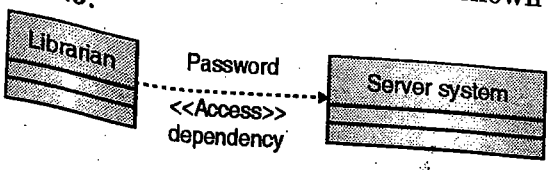


Fig. 5.2.9

Analysis Packages

Package : The package is used to assemble a collection of related classes.

For example : The analysis model of video game may include following packages :

- Environment
- Characters

In package environment we can have following related classes :

- Tree
- Landscape
- Road
- Wall
- Bridge
- Building
- Visual effect
- Scene

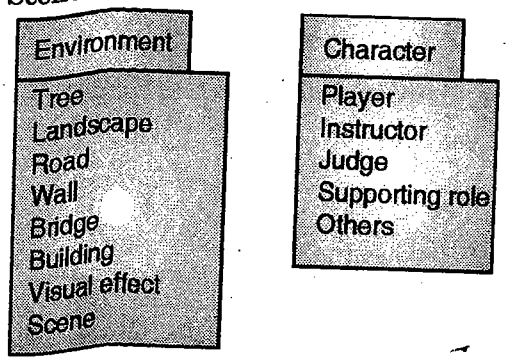


Fig. 5.2.10 : Environment and characters packages in video game

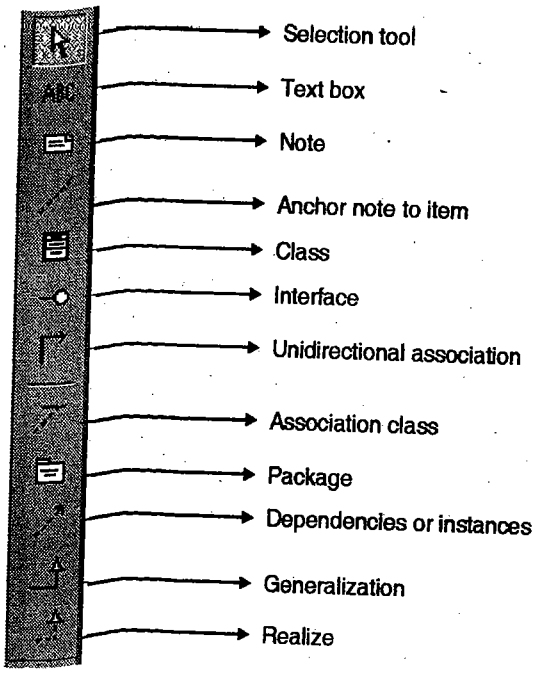


Fig. 5.2.11 : Notations in package diagram

5.3 Data Modeling

SPPU - May 15

University Question

Q. What is data modeling ? **(May 2015, 2 Marks)**

- Analysis modeling begins with data modeling. Here all data objectives that are processed within the system, their attributes and relationship between different data objects are specified.
- The relationship between different data objects is also considered in terms of number of occurrences i.e. cardinality.
- Also the relationship is optional or mandatory is represented by modality.

5.3.1 Data Objects

SPPU - Dec. 12, May 14, May 15

University Questions

Q. Discuss in short: Data objects in data models. **(Dec. 2012, May 2014, 3 Marks)**

Q. Explain term in data modeling: Data objects. **(May 2015, 3 Marks)**

- Data object is the representation of composite information. The composite information is defined as the object that has various different properties or different attributes.
- For a data object, the encapsulation of only data is present. It means here is no reference between data object and operations on the data.
- **For example,** in the table, car is a data object with properties name, colour and prize. For example data object car can be represented in tabular form as follows :

Table 5.3.1 : Tabular representation of data object

Name	Color	Prize
Honda Amaze	Silver	6 Lac
Santro	Yellow	3.5 Lac
Alto 800	Maroon	3 Lac
Hyundai i20	Blue	11 Lac

5.3.2 Data Attributes

SPPU - May 15

University Question

Q. Explain term in data modeling: Data attributes. **(May 2015, 3 Marks)**

Each data object is having set of attributes. That is data attributes define the properties of data object can have of the following characteristics :

1. Name an instance of data object.
2. Describe the instance.
3. Make reference to another instance in another table.

5.3.3 Relationship

SPPU - May 15

University Question

Q. Explain term in data modeling: Relationships. **(May 2015, 3 Marks)**

- Relationship indicates how data objects are related to one another.
- **For example:** Customer purchases the car. Here, 'purchase' is the relation.
- The relationship between data object can be represented as :

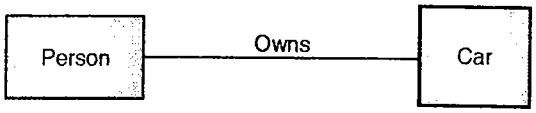


Fig. 5.3.1 : The relationship between data objects

5.3.4 Cardinality and Modality

SPPU - Dec. 12, May 14

University Question

Q. Discuss in short: Cardinality and modality in data models. **(Dec. 2012, May 2014 3 Marks)**

We represent the pictorial relationship between two objects as shown in Fig. 5.3.2.

Fig. 5.3.2

Here, the relation shows that object X is related to object Y. But, this relation does not

provide enough information for "how many occurrences of object X are related to how many occurrences of object Y". Hence such types of details are provided by cardinality and modality.

Cardinality

- Cardinality specifies number of occurrences of one object related to number of occurrences of another object.
- In other words, cardinality refers "how many occurrences of object one object are related to how many occurrences of another object".
- That is the maximum number of object relationship is represented by cardinality. Cardinality can be expressed as :

One to one (1 : 1)

It implies that one occurrence of object one is related to one occurrence of another object.

For example: An engineering college is having only one principal.

One to many (1 : n)

It implies that one occurrence of object one is related to many occurrence of another object.

For example : One class may have many students.

Many to many (m : n)

It implies that many occurrence of object one is related to many occurrence of another object.

For example : An uncle may have many nephews while a nephew may have many uncles.

Modality

A modality of relationship is zero if occurrence of relationship is optional and modality of relationship is 1 if occurrence of relationship is mandatory (i.e. compulsory). Thus, the modality specifies the minimum number of relationship.

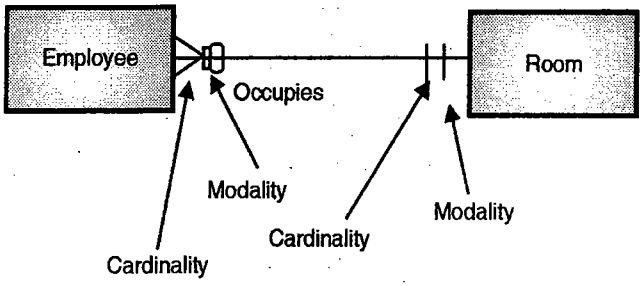


Fig. 5.3.3 : Relationship showing data objects with maximum and minimum number of occurrences

The symbol for showing the number of occurrences is :

Here, the relationship shows that "Exactly one (maximum 1 and minimum 1) room is occupied by zero or many (maximum many and minimum 0) employees or the relationship may be read as "zero or many employees occupy exactly one room".

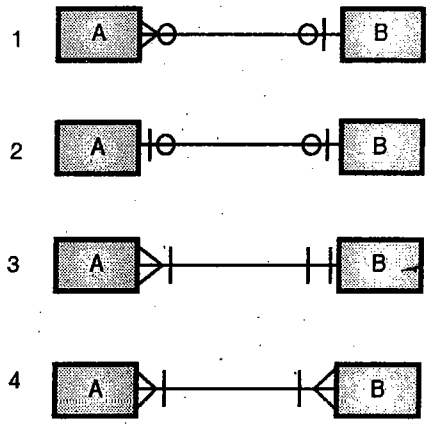


Fig. 5.3.4 : 1 to 4 showing relationship between different objects with different combinations of cardinality and modality

- Fig. 1: Cardinality is n : 1 and both roles are optional.
- Fig. 2: Cardinality is 1 : 1 and both roles are optional.
- Fig. 3: Cardinality is n : 1 and both roles are mandatory.
- Fig. 4: Cardinality is m : n and both roles are mandatory

Example of entity relationship diagram

The Fig. 5.3.5 shows the entity relationship diagram including cardinality and modality. Now, each relation can be read with maximum

and minimum number of occurrences of that object.

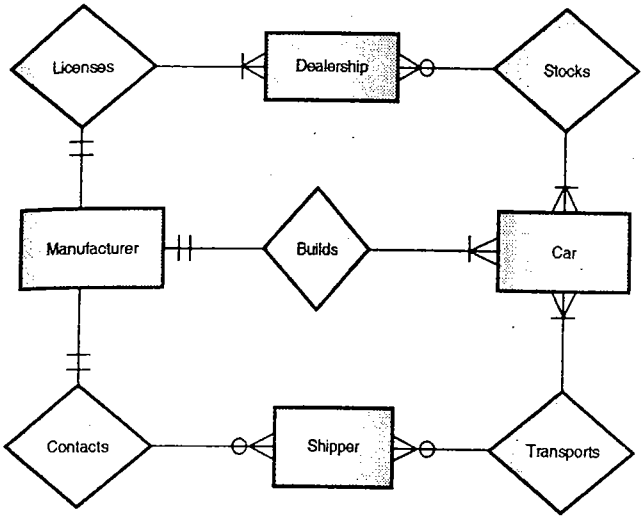


Fig. 5.3.5 : Entity Relationship diagram

For example

From the Fig. 5.3.5 we can read the relations with number of occurrences as follows :

- A manufacturer builds a car.
- A manufacturer contacts to zero or many shippers.

5.4 Data and Control Flow Model

5.4.1 Flow-Oriented Modeling

They provide necessary information that how data objects are transformed by processing the functions. Flow oriented elements are :

1. Data Flow Diagrams
2. Control Flow Diagrams
3. Control Specifications
4. Process Specifications

5.4.1.1 Data Flow Model

- Data Flow Diagram (DFD) is also called as 'Bubble chart' is a graphical technique, which is used to represent information flow, and transformers those are applied when data moves from input to output.
- DFD represents system requirements clearly and identify transformers those becomes programs in design. DFD consists of series of bubbles joined by lines.
- DFD may be further partitioned into different levels to show detailed information flow e.g. level 0, level 1, level 2 etc.

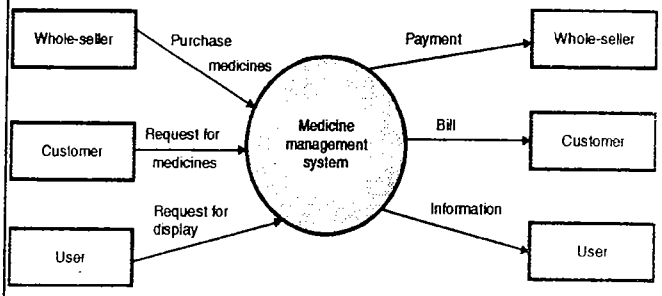


Fig. 5.4.1 : Data flow diagram for medicine managements system

- DFD focuses on the fact 'what data flow' rather 'how data is processed'.

Example of data flow diagram

The data flow diagram developed for **medicine management system** includes the external entities : Whole-seller, Customer and User. The DFD shows following requirements with proper flow of data from one entity to other.

- The medicines are purchased from whole-seller.
- The customer can request for the medicines in medicine store.
- User can request for display of the current status of the items in medicine.
- When medicines are purchased from whole-seller, the system generates the payment for whole-seller.
- When medicines are sold to customer, the system generates the bill for customer.
- The system also presents the required information to the user as per his requirements.

Developing level 1 DFD

- DFDs are used to represent information flow, and the transformers those are

applied when data moves from input to output.

- To show the data flow with more details, the DFD is further extended to level 1, level 2, level 3 etc. as per the requirements.
- The typical value for the DFD is seven. Any system can be well represented with details up to seven levels.
- The level 0 and level 1 DFD for the 'Food Ordering System' in the restaurant' are shown in Figs. 5.4.2 and 5.4.3.

Notations used in DFD

- The circle represents the process.
- The rectangle represents the external entity like customer, whole-seller etc.
- The labelled arrows indicate incoming and outgoing data flow.
- The open rectangle shows the database or file.

DFD for food ordering system

Here customer, kitchen and restaurant managers are external entities.

- The 'food order system' accepts the food order from the customer and forwards the order to the kitchen.
- When the service is provided to the customer, the system generates the bill.
- A copy of customer bill can be submitted to the manager as a part of restaurant management report.

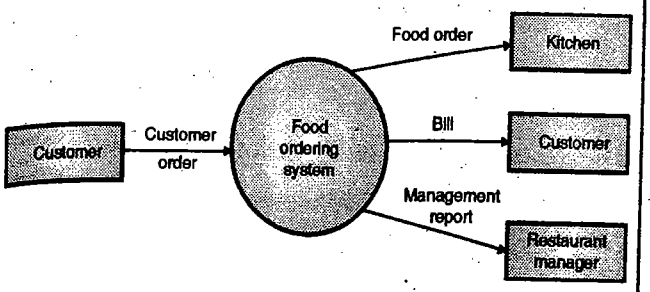


Fig. 5.4.2 : Level 0 DFD for 'food ordering system' in restaurant

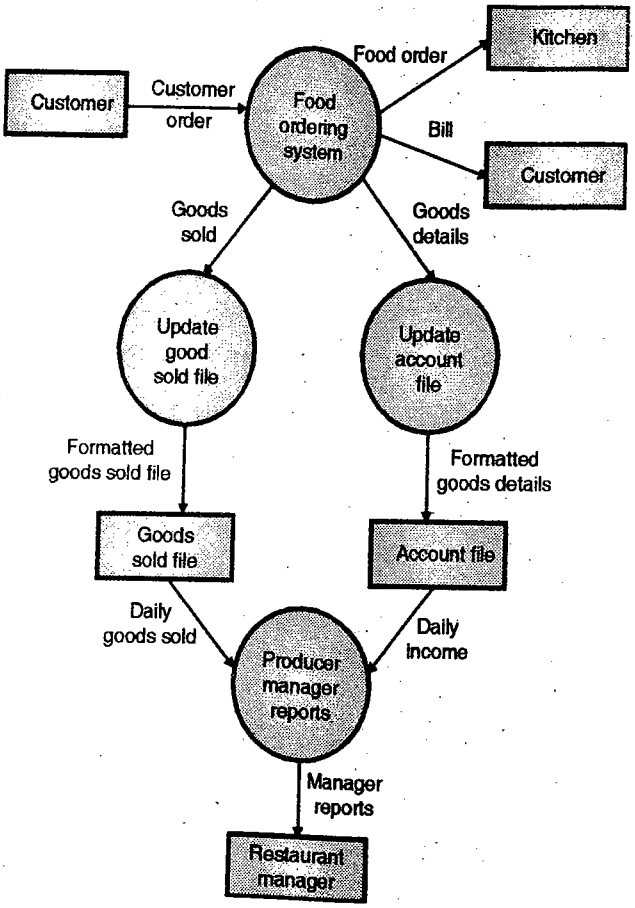


Fig. 5.4.3 : Level 1 DFD for 'food ordering system' in restaurant

This level 0 DFD can be extended to level 1 DFD to show more details showing exact data flow and processes (i.e. transformers).

5.4.1.2 Control Flow Model

The large class of applications having following characteristics requires control flow modeling :

- The applications those are driven by the events rather than data.
- The applications those produce control information rather than reports or displays.
- The applications those process information in specific time.

The control item or event is implemented as Boolean value. For example, true or false, on or off, 1 or 0.

5.4.1.3 Control Specifications

The control specification represents the behaviour of the system. The behaviour of the system can be represented using 'state

transition diagram' which is also called as 'state chart diagram'.

A state chart diagram

A state chart diagram shows the state machine that consists of states, transitions, events and activities. This diagram shows how system makes the transition from one state to another state on occurrence of particular event.

State

A state of an object is defined as the condition that is satisfied in the life of an object and once the condition is satisfied, the object performs some activity and it waits for some events to occur.

Event

An event causes the system to make transition from one state to another. The notations used in state chart diagram are :

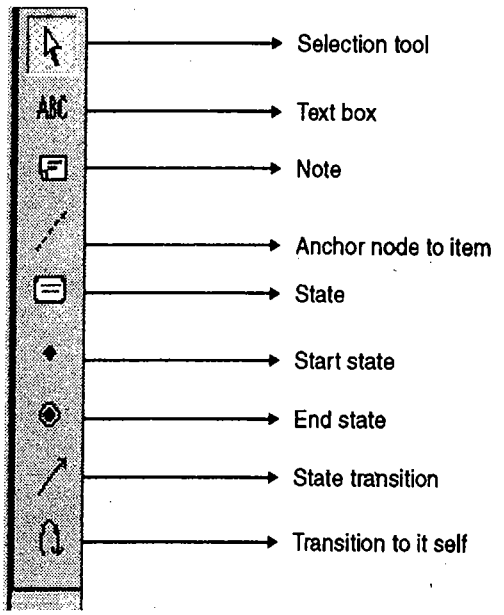


Fig. 5.4.4 : Notations in state chart diagram

Example of state chart diagram

Problem statement : To develop the state chart diagram for 'washing machine':

Explanation

- First, fill the water in washing machine. Then take detergent in. Now, motor is in running state.
- The 'motor running' state can go in ' motor stops' state in one of the two cases :
  - When user of washing machine will press the stop switch or
  - Time expires
- When 'washing process' is completed, the transitions will take place for 'drying process'. When drying process is going on the machine reaches to end state in one of the two cases :
  - When user of washing machine will press the stop switch or
  - Time expires

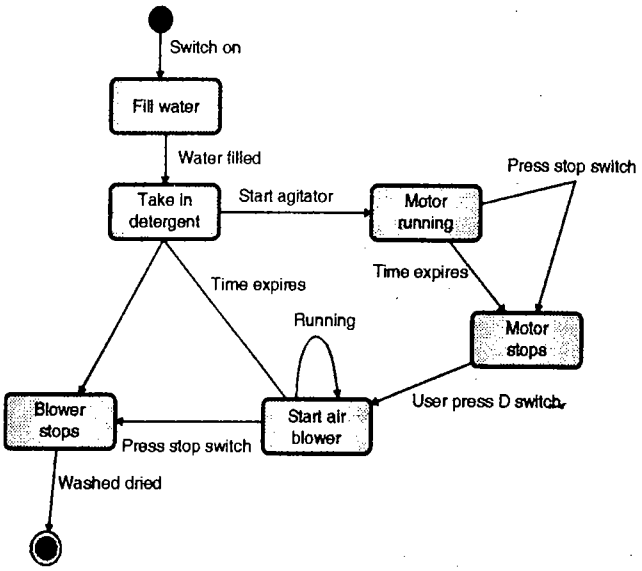


Fig. 5.4.5 : A state chart diagram for washing machine

5.4.1.4 Process Specifications (PSPEC)

The process specification is used to describe all flow model processes. The content of process specifications include Program Design Language (PDL). The PDL is also called as **'pseudo code'** or **'Structured English'**. PDL uses the vocabulary of English and syntax of some other language. PDL looks like the modern programming language but it cannot be compiled. PDL processors are used to convert PDL into graphical representation.

Example of process specification

Consider following part of DFD with process 'analyze triangle'.

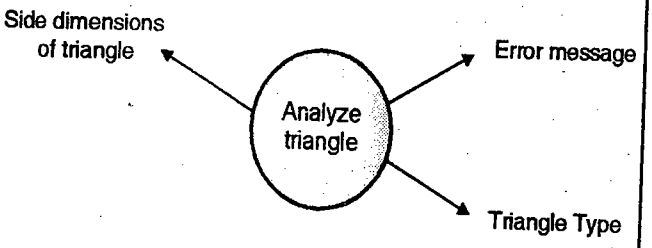


Fig. 5.4.6

5.5 Behavioral Modeling using State Diagrams

SPPU - May 13

University Question

Q. Explain in detail UML diagrams stating purpose and applicability: State diagram. **(May 2013, 3 Marks)**

- The main task of behavioural model is that it indicates responses of software application due to occurrence of external events. In order to create a behavioural model, the system analyst should follow the steps as mentioned below :
  - First evaluate all the use-cases. This evaluation will help to understand the complete sequence of interaction within the system.
  - Identify each event and see how these events are related to the specific classes.
  - Next, create the sequence for the use-cases that are considered.
  - Now build the state diagram for the system under consideration.
  - Finally review the complete model to evaluate it for accuracy and consistency.
- All these steps are discussed in detail in the following section.

5.5.1 Identifying the Events with Use-Cases

- We know that use cases represent the sequence of various activities occurring in a system.
- But generally any event occurs whenever there is an information exchange between the system and actor.
- It is important to note that the event is not an information. In this scenario, the event has not been changed but the information has been changed.
- For an example consider any software application where a four digit password is required to validate the user authentication. The user uses the key pad to enter four digit password and this password is compared with the one stored in the system. If the password is incorrect it will not allow using the application.
- Once the password is correct, it will allow users to use the system. This portion of use case scenario indicates the various events occurring.
- If password is correct then only it will have the impact on the information and control flow of the system in use.

5.5.2 Create the Sequence for Use-Case

These diagrams are used to represent the behaviour of the system. These diagrams show how events cause transition from object to object. Once the events are identified from the Use-cases, sequence diagrams can be created.

Notations used in sequence diagram

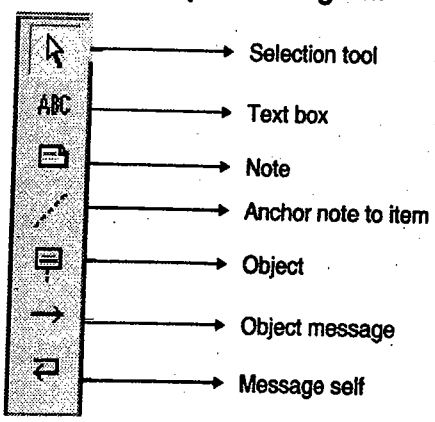


Fig. 5.5.1 : Notations used in Sequence Diagram

Example of sequence diagram

- To develop the sequence diagram for **'Computer Based System'**.

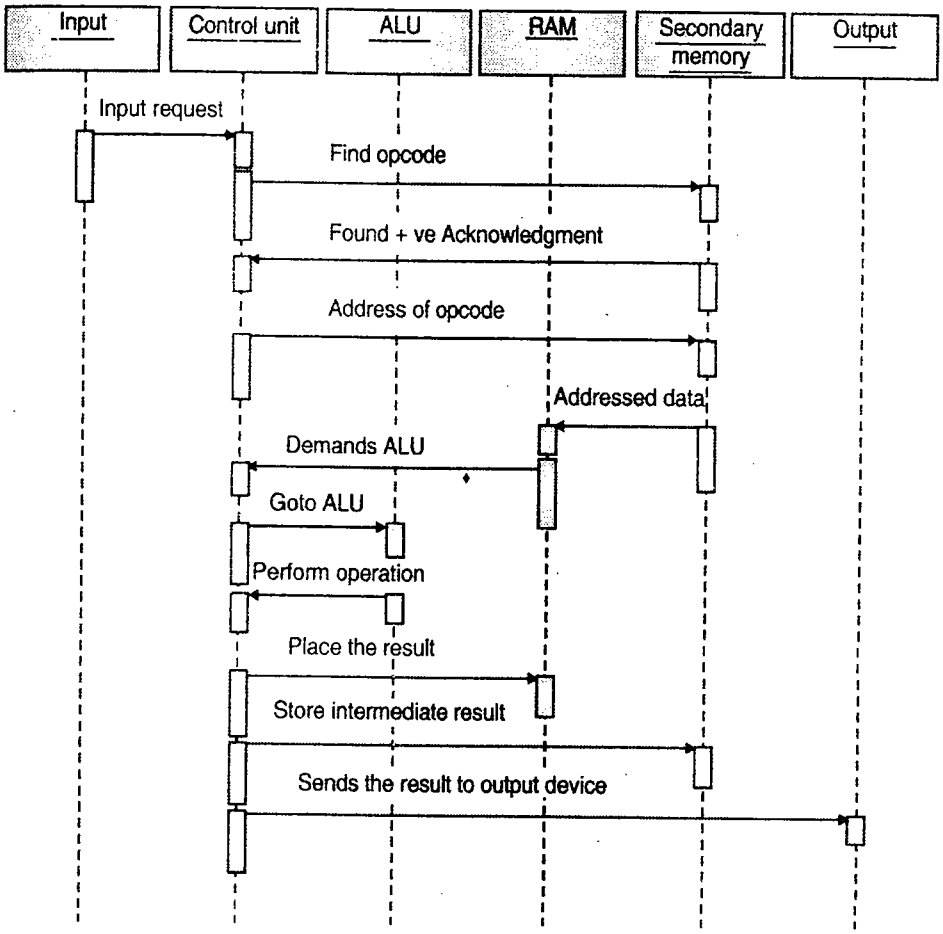


Fig. 5.5.2 : Sequence diagram for Computer Based System

5.5.3 State Machine Diagram with Orthogonal States

- One of the important components of bahavioural model is UML state machine diagrams that represent the active states and events between the active states.
- A state machine is defined as a behaviour that is exhibited by the sequences of states of an object during its lifetime. These states exist due to occurrence of events.
- The state machines are modelled by the dynamic aspects of a system. During the modeling phases, the state machine specifies the lifetime of the instances of a class. It also specifies the lifetime of a use case may the whole system.
- All these instances respond to the events like operations, signals or the pastime. Whenever an event occurs, any of the activity may take place based on the current state of an object.
- The activity in state machine is defined as an ongoing non-atomic execution within it. These activities cause some action that result in changing the state of an object or the model. The actions may be defined as executable atomic computations.
- The state of an object satisfies any condition during its lifetime and due to this it performs some activity and wait for some events to occur.

- We see state machine by two points of view :
  - o By putting emphasis on flow of control in different activities and
  - o By putting emphasis on the states itself by using state chart diagrams
- A well structured state machine is very similar to a well structured algorithm in the sense that it adaptable to environment, it is simple and efficient and it is easy to understand.
- A state machine is generally used to model the behaviour of the elements of a class, use-case or the complete system.
- A state machine is viewed in two different ways :
  - o By using activity diagrams or
  - o By using the state machine diagrams
- Here the activity diagram focuses on the activities occurring within an object and state machine diagrams focus on the behaviour of an object.
- The UML diagrams illustrate the graphical representation of states, its transitions, events and their actions as depicted in Fig. 5.5.3.

5.5.3.1 Orthogonal States

- Following diagram shows that the orthogonal regions are entered in the state machine diagram. Basically the state with two or more than two regions is called as orthogonal state.
- A transition is a path that shows the change of state is occurring. In the following diagram two regions are represented. Both the regions finish in parallel and reach the final node i.e. S6. This is an example of a state machine with orthogonal states.

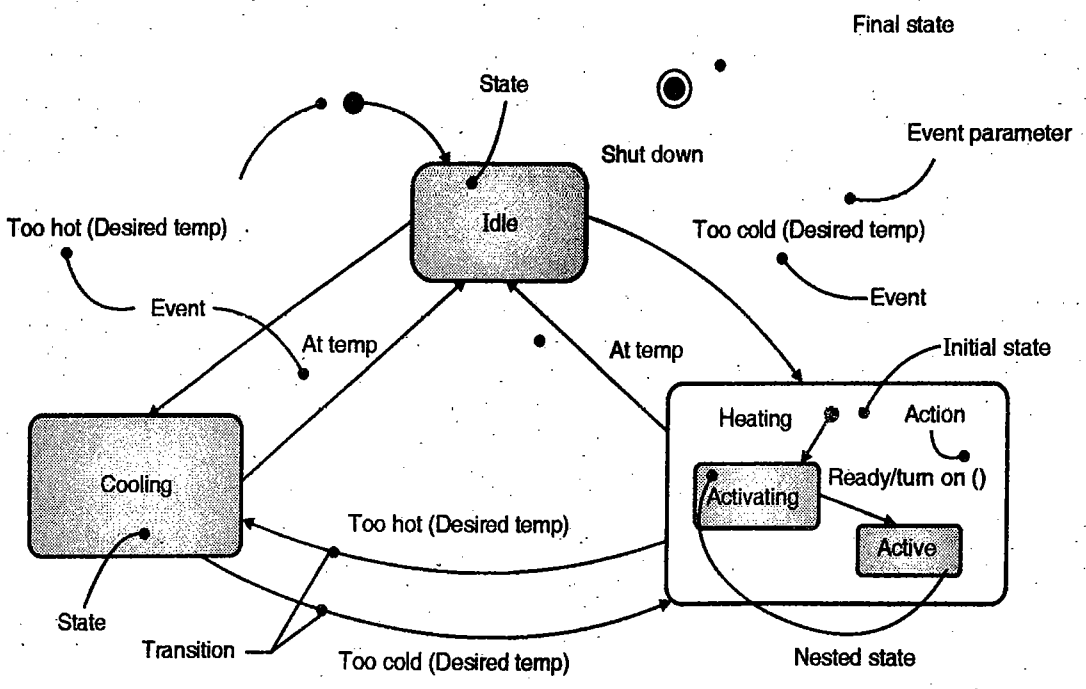


Fig. 5.5.3 : State machines

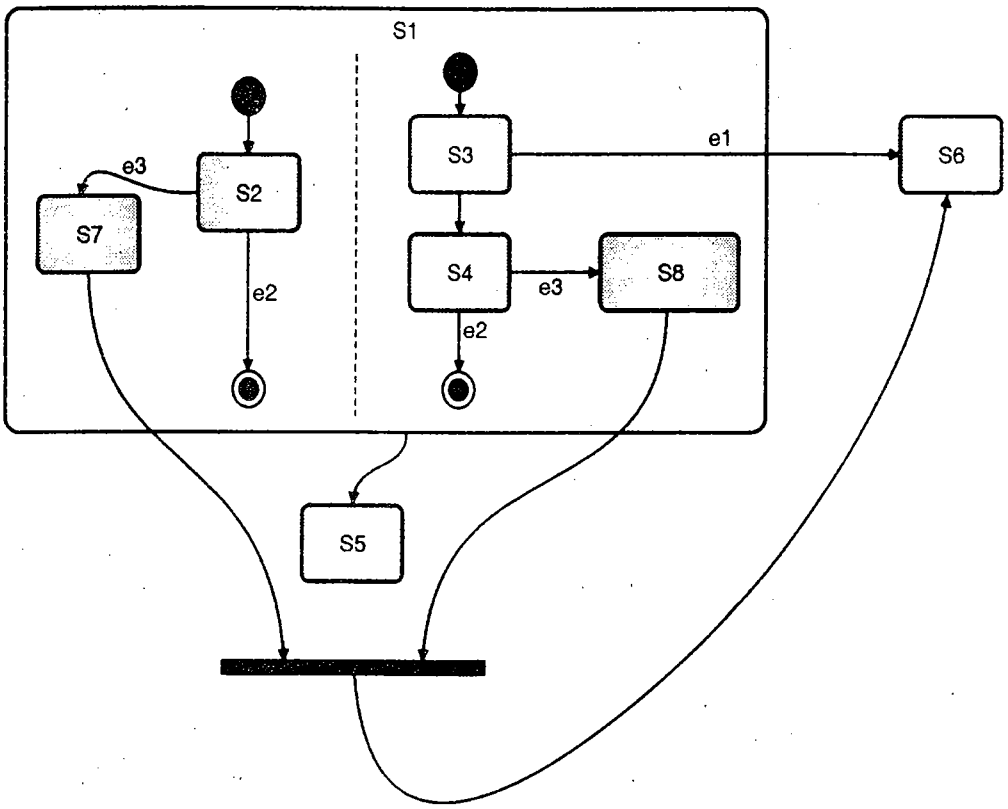


Fig. 5.5.4 : State machine with orthogonal state

5.6 Requirements Modeling for WebApps

- Requirement modeling is important part of the software engineering process. Following are the tasks for requirement engineering :
  1. Requirement elicitation
  2. Requirement specification or
  3. Requirement validation
- All these tasks ensure the quality of the software product under development. The development of web applications involves some more tasks in addition to these tasks. The additional requirements are considered for navigational and multimedia aspects.
- The requirement engineering in traditional systems is considered as an important phase in the development process. This is most time consuming phase and it is supposed to be most expensive phase to repair the errors. Most of the errors are due to improper and inadequate requirement engineering. Many techniques are available to handle these issues. Some are interviewing customers, frequent meetings with customers and Formal Technical Reviews (FTR) etc.
- In contrast, all these techniques are poorly applied in web based developments. The web applications require more detailed requirement engineering as compared to traditional developments processes. Since the large number of stakeholders are using web applications. The requirement engineering is always an iterative process.
- All the developers find the development very difficult since there is always a pressure to deliver very quality web based applications and also that should be economical and within the time limits.

5.6.1 Requirements Engineering Techniques

- The requirement is defined as condition that must be fulfilled by the application and should satisfy the specification and conditions elicited by the customers.

- The requirements must be correctly defined by the customers and it should be consistent and verifiable. This process is called as requirement engineering in which customers requirements are understood and validated.
- The development team should obtain all the information about problem domain and web application's requirement specifications.
- Even though the requirement elicitation and specification is a complex process, for the developers it is mandatory to identify all the functionalities that should satisfy customers.
- There is lack of standard processes that should support requirement engineering. But the developers should guarantee the best quality of the product.
- The requirement engineering is iterative in nature. The requirement engineering process for web applications consists of following three main activities :
  - Elicitation
  - Specification and
  - Validation of requirements
- Following Fig. 5.6.1 shows the process of requirement engineering for web applications :

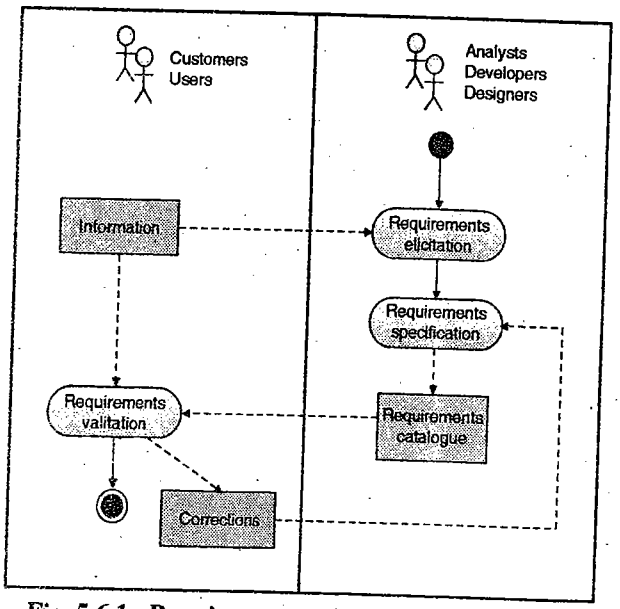


Fig. 5.6.1 : Requirement engineering for WebApps

- The process begins with requirements elicitation by the users or the customers. The developer's team collect all the information from various sources like interviews, legacy software applications, and documents etc.
- After requirements elicitation is finished, requirement specification is done and the requirement catalogue is prepared.
- Finally the requirements validation is started to verify the requirements that is fulfilled by the application and satisfied by the user. In this phase, inconsistencies are also found out and any undefined requirements are also found out and settled.
- As the requirement engineering process is iterative in nature, in complex projects it is executed several times to refine the requirements.

5.6.2 Requirements Elicitation

The development team performs the requirement elicitation process which is a complex one. Following are some most commonly and effective techniques used to capture requirements :

- **Interviews** are most commonly used techniques for discovering the requirements from the users or the customers. The development team should follow some guidelines for interviewing. Following are four steps for interviewing :
  - Identify the correct stakeholders
  - Prepare for interview (like what questions to be asked)
  - Interview and
  - Documentation
- **JAD (Joint Application Development)** is technique in which all the stakeholders of the project participate, i.e. analyst,

designers, developers, administrators and customers all sit together for several sessions for finalizing the specification.

- **Brainstorming** is meeting very much similar to JAD but the number of participants (i.e. stakeholders) should not exceed. It consists of non-evaluated ideas and information. Brainstorming always gives the better ideas of system requirements.
- **Concept mapping** is a process in which concept maps are built. The concept maps are the graphs consisting of vertices and edges that represent concepts and relationships between the concepts respectively.
- **Sketching and storyboarding** is a technique used in Web applications. It consists of representation of the different user interfaces e.g. sketches. It gives idea about the requirements.
- **Use case modeling** is used to define requirements. A use case model has following important elements :
  - o Actors,
  - o Use cases and
  - o Relationships between them.
- It defines all the functional requirements.
- **Questionnaire and checklist** consists of preparing a document with number of questions and short and concrete answers (i.e. checklist).

5.6.3 Requirements Specification

The most widely requirements specification techniques are described as follows :

- **Natural language.** In this requirements are narrated in natural language without any rules.
- **Glossary and ontology** used to define the terms used in development project since different stakeholders are from different background and they work together.
- **Templates** are used to write the objectives and requirements using the natural language.
- **Scenarios** are the descriptions of the characteristic of the application under development.
- **Prototypes** are used to better understand the requirement and refined accordingly.
- **Use case modeling** is used define the requirements.
- **Formal description** is also a group activity in which descriptions are specified by using formal languages.

5.6.4 Requirements Validation

Following are some techniques illustrated that are used in requirement validation :

- **Review** is a technique in which reading and correcting the requirements documents and models and verification of any inconsistencies are done.
- **Audit** is a cross checks of results presented in review.
- **Traceability matrix** is a correspondence established between different objectives that should cover each of the requirements.
- **Prototyping for validation** is used to present partial functional requirements for quick reviews and feedbacks from the users. This is not the final product.

5.6.5 Requirements in Current Web Methodologies

The requirements engineering used in current web methodologies can classified as :

- Functional requirements and
- Non-functional requirements

5.7 Real Life Application Case Study

- The development of Web applications is an important focus of the modern information

enabled organization – whether the Web application development is in-house, outsourced, or purchased as 'commercial-off-the-shelf' (COTS) software.

- Traditionally Web application development has been delivered via the dominant waterfall system. The waterfall system relies upon well-defined governance structures, linear phases, gating, and extensive reporting and sign-off documentation.
- An increasing number of development stakeholders criticize the waterfall system for web application development. The criticisms include a disproportionate focus on governance and process at the direct expense of flexibility and, most importantly, reduced productivity. One consequence of these criticisms is the increasing adoption of Web application development via agile-system methods.
- This agile-system approach centers upon smaller design teams, fewer development phases, and shorter development time tables.
- This case study examines the implementation of the agile-system approach as used by a Small-to-Medium Enterprise (SME) software developer. The case study data collection involves interviews and observations across three different SME sources: project managers, Web application programmers, and customers.
- The case study analysis synthesizes the experiences of these managers, programmers and customers to produce an overall assessment of the usefulness of Web application delivery via agile-system methods.
- The objective of this research was to examine the use of agile methods within a SME software developer, and to gain an understanding of the enabling and limiting factors associated with the usage of these agile methods.
- The SME software developer deploys a project team of seven staff (plus one consulting customer representative) to produce software that is best characterized as web driven client service interfaces to back end database services.
- The data collection within this case study was conducted via qualitative research methods. A series of primary and secondary personal interviews were conducted over the four months of the case study with the SME project manager and several key project stake-holders.
- The goals of this case study research were to describe how Scrum and XP practices had been tailored for use within a SME software developer, and to assess the efficiencies and risks of this tailored use.

5.8 Software Requirement Specification (SRS)

- Basically SRS or software requirement specification is an official document or the statement of what the system developers should implement.
- It includes both :
  - o System requirement.
  - o User requirement.
- The SRS has a set of users like senior management of the organization, engineers responsible for developing the software.

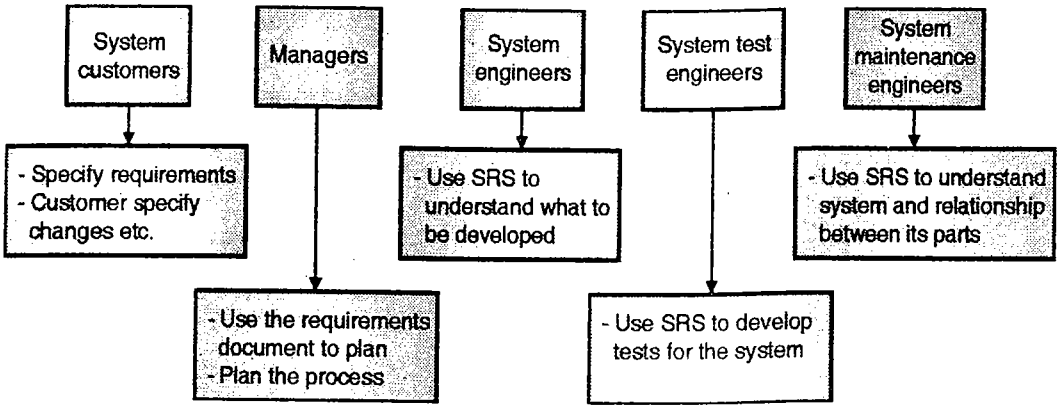


Fig. 5.8.1 : Users of SRS

- The details which are included in SRS depend on the type of system that is being developed and the type of the development process.
- A number of large organisations, such as IEEE have defined standards for software requirements document.
- The most widely used standard is IEEE/ANSI 830-1998. This standard suggests the following structure for requirements document :
  1. Introduction
     - o Purpose of SRS
     - o Scope of product.
     - o Definitions, acronyms, abbreviations.
     - o References
     - o Overview
  2. General description
     - o Product perspective
     - o Product Functions
     - o User characteristics
     - o General constraints
     - o Assumptions and dependencies
  3. Specific requirements
  4. Appendices
  5. Index.
- SRS is very useful when an outside contractor is developing the software system.
- For business system, where requirements are unstable, SRS play an important role.

5.8.1 Writing Software Requirements Specifications

- Proper software requirement gathering is very important at the start of the software development. Generally it is done by professional software developers.
- Document is created in the requirement analysis which is generally referred as SRS or software requirement specification document.
- It is first project deliverable.
- SRS records the end user needs in certain format. This is SRS is needed when actual software development process starts.

5.8.2 What is a Software Requirements Specification?

- Before starting software development requirement is captured from the end users or clients or customer. This requirement is written in certain format which is called as SRS. Generally this document is created before starting the development work.
- It signifies both developers and client understood what should be implemented in the software.
- All capabilities and functionalities are stored in SRS.
- Requirement document needs in every steps of software development.

- Analyst uses this document for designing the software. Developer uses this document during the coding whereas software tester uses this document to check the functionalities of the software.
- All remaining project documents are depends upon requirement document because of which it is referred as parent of all document.
- It contains functional and non functional requirements.

Good SRS have accomplishes following goals

- It gives feedback to customer.
- The large problem can be divided into different small components.
- It acts as input to design which is part of design specification.
- It acts as product validation check.

5.8.3 What Kind of Information Should an SRS Include?

Following things are part of SRS :

- Interfaces
- Functional capabilities
- Performance levels
- Data structures/Elements
- Safety
- Reliability
- Security/Privacy
- Quality
- Constraints and limitations

5.8.4 SRS Template

- Different existing SRS templates can be used in writing SRS documents.
- We can select the template which matches our requirement.
- Template will just acts as guiding principles for writing template. Proper changes need to be done whenever necessary in template.
- Following table shows SRS outline :

Table 5.8.1 : A sample of a basic SRS outline

1. **Introduction** : It contains different details like purpose of software, conventions used, target audiences, extra information, SRS team members, references.
2. **Overall Description** : Overall information of the project is given in this sections. It includes perspective, functions, different user classes, working environment, design constraints, assumptions about the software.
3. **External Interface Requirements :** It contains external interface information which includes user interface, hardware interface, software interfaces, communication protocols, etc.
4. **System Features :** Which system features needs to add is given in system features. It includes different features, features description, priority, action result, functionalities, etc.
5. **Other Nonfunctional Requirements** : Non functional requirement of the project is given in this section. It includes performance requirement, safety concerns, security constraints, quality factors, documentation, user documentation.
6. **Other Requirements** It includes Appendices, glossary of the software, etc.

5.8.5 Characteristics of an SRS

Review Question

Q. Explain different characteristics of Software Requirement Specification ?

- **Complete :** All the project functionalities should be recorded by SRS.
- **Consistent :** SRS should be consistent
- **Accurate :** SRS defines systems functionality in real world. We need to record requirement very carefully.
- **Modifiable :** Logical and hierarchical modification should be allowed in SRS.
- **Ranked :** Requirement should be racnked using different factors like stability, security, ease or difficulty.
- **Testable :** The requirement should be realistic and able to implement.
- **Traceable :** Every requirement we must be able to uniquely identify.
- **Unambiguous :** Statement in the SRS document should have only one meaning.
- **Valid :** All the requirements should be valid.

❑❑❑

UNIT III

CHAPTER 6

Project Planning

Syllabus

Project initiation, Planning Scope Management, Creating the Work Breakdown Structure, Effort estimation and scheduling: Importance of Project Schedules, Estimating Activity Resources, Estimating Activity, Durations, Developing the Schedule using Gantt Charts, Adding Milestones to Gantt Charts, Using Tracking, Gantt Charts to Compare Planned and Actual Dates, Critical Path Method, Program Evaluation and Review, Technique (PERT) with examples. Planning Cost Management, Estimating Costs, Types of Cost Estimates, Cost Estimation Tools and Techniques, Typical Problems with IT Cost Estimates.

Syllabus Topic : Project Planning

6.1 Introduction to Project Planning

Review Questions

Q. Why Project Planning is important ?

Q. List the activities in the software project planning.

Q. What is the objective of project planning ?

- The project planning is an important activity performed by the project managers in order to estimate the following entities for their optimum use :
  - o The resources required for the project under development.
  - o The cost of the project, and
  - o The schedule for in-time completion of the project
- All these estimates are made in the stipulated amount of time in the starting of the project itself. Later on these estimates can be updated time to time. In order to achieve the objectives of project planning, following activities are carried out :
  - o Defining software scope
  - o Estimating resources

6.1.1 Defining Software Scope

Software Scope

- A software project manager is normally confused with a condition in the beginning of a software engineering project i.e. quantitative estimates and an organized plan. They are required and necessary but proper information is not available.
- A detailed analysis of software requirements may provide all the necessary information for estimates, but analysis often takes too much time to complete, even weeks or months. Requirements may be changing regularly as the project proceeds.
- Therefore, the developer and the manager should examine the product and the problem. It is intended to solve at the beginning of the project itself. At a minimum, the scope of the product must be established and bounded.
- The first software project management endeavor is the finding out of software

scope. Scope can be defined by answering the following simple questions.

- **Context :** How the software system can be developed to fit into a large system or some business context and what are the constraints that should be considered as a result of the context considered ?
- **Information objectives :** What data objects are produced as output through the software developed from customer's point of view ? And what data objects will be used for the input ?
- **Function and performance :** What are the functions that the software should perform in order to translate input into output ? Is there any special performance characteristic that is supposed to be addressed ?

- There should be a clear and comprehensive software project scope defined and that should be understandable at all the levels (i.e. the management and technical levels).
- Generally the software scope describes the following :
  - Performance
  - Function
  - The data and control used to define the constraints
  - Reliability and
  - Interfaces.
- Functions are described in the statement of scope are assessed and evaluated to provide details in estimation. The cost and schedule are two important parameters and they are functionality oriented.

6.1.2 Obtaining Information Necessary for Scope

- In start of a development process, it is always good and necessary to define the scope of the project. The frequent communication between the customer and developer builds the foundation for defining scope properly. The frequent meetings between the customer and developer are very important to bridge the gap in all sorts of communication.
- Following are the set of questions that focuses on the goals and objectives :
  - Know the person behind the request for this work.
  - See who will use the solution?
  - What will be the commercial benefit of a solution?
  - Is there any other resource of solution?
- These questions clear the understanding of problems and customers can raise their doubts.

Syllabus Topic : Project Initiation

6.2 Project Initiation

Review Question

Q. Explain Project Initiation in details.

- The Project Initiation Phase is the first phase in the Project Management Life Cycle, as it involves starting up a new project. We start a new project by defining its objectives, scope, purpose and deliverables to be produced.
- We also hire our project team, setup the Project Office and review the project, to gain approval to begin the next phase.
- Overall, there are six key steps that we need to take to properly initiate a new project. These Project Initiation steps are listed below :
  - Develop a Business Case
  - Undertake a Feasibility Study
  - Establish the Project Charter
  - Appoint the Project Team
  - Set up the Project Office
  - Perform a Phase Review

6.2.1 Business Case

- A Business Case justifies the start-up of a project. It includes a description of the business problem or opportunity, the costs and benefits of each alternative solution, and the recommended solution for approval.
- The Business Case is referred to frequently during the project, to determine whether it is currently on track. And at the end of the project, success is measured against the ability to meet the objectives defined in the Business Case.

6.2.2 Feasibility Study

- A Project Feasibility Study is an exercise that involves documenting each of the potential solutions to a particular business problem or opportunity. Feasibility Studies can be undertaken by any type of business, project or team and they are a critical part of the Project Life Cycle.
- The purpose of a Feasibility Study is to identify the likelihood of one or more solutions meeting the stated business requirements.
- In other words, if you are unsure whether your solution will deliver the outcome you want, then a Project Feasibility Study will help gain that clarity. During the Feasibility Study, a variety of 'assessment' methods are undertaken. The outcome of the Feasibility Study is a confirmed solution for implementation.

6.2.3 Project Charter

- A Project Charter outlines the purpose of the project, the way the project will be structured and how it will be successfully implemented. The Project Charter describes the project vision, objectives, scope and deliverables, as well as the Stakeholders, roles and responsibilities.
- The Project Charter defines the vision and boundaries for the project, as well as the high level roadmap. In addition, the Project Charter also defines the scope of the project, within which the deliverables are produced.

6.2.4 Project Team

- A Project Job Description defines the objectives and responsibilities of a particular role on a project.
- A Project Job Description should be completed every time a new role is identified. The Project Job Description should clearly state the objectives and responsibilities of the role and where it fits within the organizational structure.

6.2.5 Project Office

- The Project Office Checklist lists everything you need to do, to set up a Project Management Office. A Project Management Office is the physical premises within which project staff (e.g. the Project Manager and support staff) reside.
- The Project Office also contains the communications infrastructure and technologies required to support the project.

6.2.6 Phase Review

- A Project Review is an assessment of the status of a project, at a particular point in time. The first time in the project life cycle that a project review is undertaken is at the end of the first project phase, called "Initiation".
- During this project review, a decision is made as to whether or not the team has met the objectives and is approved to proceed to the next project phase, being the "Planning" phase.

Syllabus Topic : Planning Scope Management

6.3 Planning Scope Management

Review Questions

Q. Explain Planning Scope.

Q. Write a short notes on Planning Scope Management.

- A software project manager is normally confused with a condition in the beginning of a software engineering project i.e. quantitative estimates and an organized plan. They are required and necessary but proper information is not available.
- A detailed analysis of software requirements may provide all the necessary information for estimates, but analysis often takes too much time to complete, even weeks or months. Requirements may be changing regularly as the project proceeds.
- Therefore, the developer and the manager should examine the product and the problem. It is intended to solve at the beginning of the project itself. At a minimum, the scope of the product must be established and bounded.
- The first software project management endeavor is the finding out of software scope. Scope can be defined by answering the following simple questions:
  - **Context :** How the software system can be developed to fit into a large system or some business context and what are the constraints that should be considered as a result of the context considered ?
  - **Information objectives :** What data objects are produced as output through the software developed from customer's point of view ? And what data objects will be used for the input ?
  - **Function and performance :** What are the functions that the software should perform in order to translate input into output ? Is there any special performance characteristic that is supposed to be addressed ?
- There should be a clear and comprehensive software project scope defined and that should be understandable at all the levels (i.e. the management and technical levels).
- Generally the software scope describes the following :
  - Performance
  - Function
  - The data and control used to define the constraints
  - Reliability and
  - Interfaces.
- Functions are described in the statement of scope are assessed and evaluated to provide details in estimation. The cost and schedule are two important parameters and they are functionality oriented.

6.3.1 Obtaining Information Necessary for Scope

- In start of a development process, it is always good and necessary to define the scope of the project. The frequent communication between the customer and developer builds the foundation for defining scope properly. The frequent meetings between the customer and developer are very important to bridge the gap in all sorts of communication.
- Following are the set of questions that focuses on the goals and objectives :
  - Know the person behind the request for this work.
  - See who will use the solution?
  - What will be the commercial benefit of a solution?

- Is there any other resource of solution?

- These questions clear the understanding of problems and customers can raise their doubts.

6.3.2 Feasibility

- After defining and identifying the scope of the project, the developer should look at the scope and should estimate the feasibility whether the project is feasible or not.
- Otherwise, after spending lot of time working on the project, it is found that the scope is not feasible to develop. Hence it is always necessary to check the feasibility in the starting of the development process itself to avoid the losses.

6.3.3 A Scoping Example

- The frequent communication between the customer and developer is always useful in defining the functions, data and control, performance and the constraints and all the related information.
- The person planning the project looks into the details of statement of the project and derives all the necessary information. This particular process is called as decomposition and it has the following functions :
  - Read the input barcode
  - Read the tachometer reading
  - Database look-up should be done
  - Produce the control signal
  - Maintain all the record
- The planner (the person who plans) interacts with all the elements of computer-based system and he considers all the complexities in each of the interfaces in order to find any effect on the cost, resources and schedule.

Syllabus Topic : Creating the Work Breakdown Structure

6.4 Creating the Work Breakdown Structure

Review Question

Q. Explain Work Breakdown Structure.

- Scheduling of different engineering activities can use the available project scheduling tools and techniques.
- Following are some project scheduling tools and techniques :
  - PERT (Program evaluation and review technique)
  - CPM (Critical path method)
- These are two project scheduling methods that can be applied to the software development process.
- Both tools use the data and information received from the earlier developments. They perform the same planning activities as performed in earlier projects. These activities are listed as follow :
  - Estimating the effort
  - Decomposing the product function
  - Selecting suitable process model, and
  - Decomposing task.
- The interdependencies among various task are also called as **work breakdown structure** (WBS). The WBS can be defined for single functions and for the whole project as well.
- Both tools allow to determine the critical path, duration of the projects, and time estimate for the individual activity, and helps in calculating the "boundary times".
- The boundary time calculations are very important in defining the software project schedules.

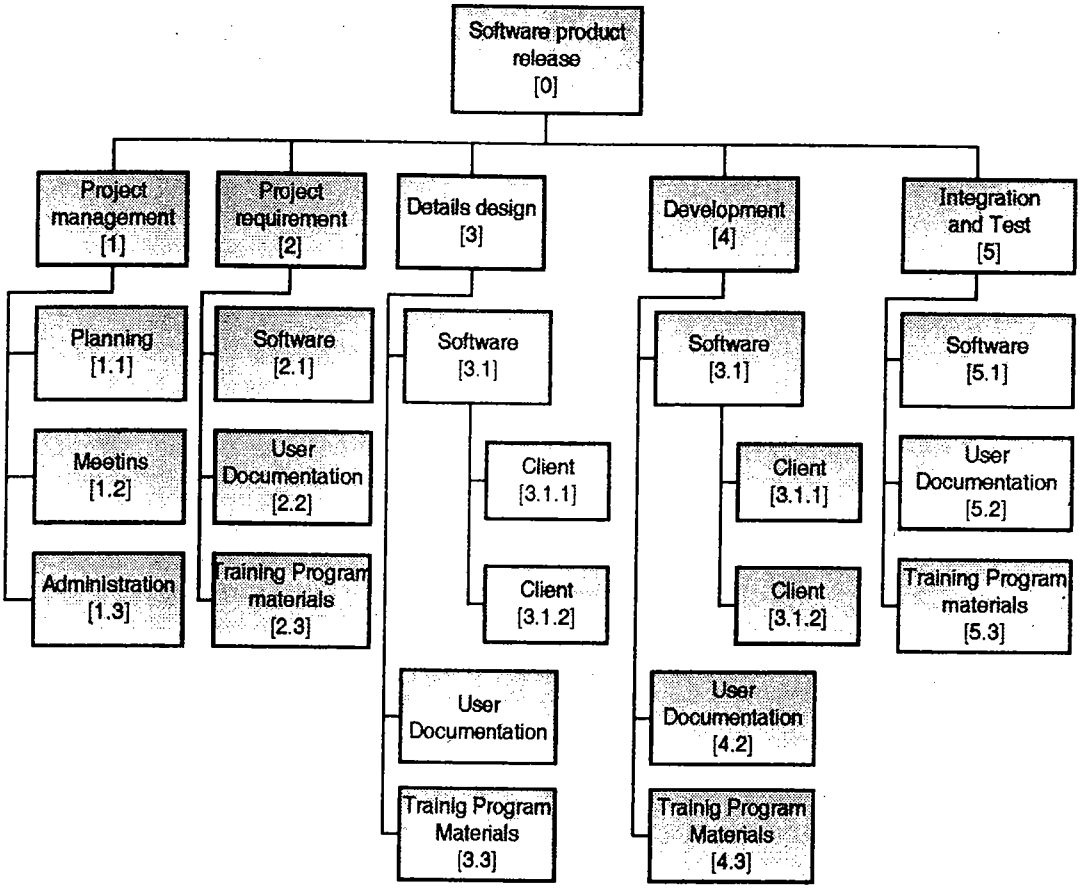


Fig. 6.4.1 : A sample Work Breakdown Structure (WBS)

6.5 Software Measurement

Review Question

Q. Explain Software Measurement.

- We have seen that software measurement can be classified in two ways:
  - **Direct measures** of the software process and product (e.g., Lines of Code (LOC) produced).
  - **Indirect measures** of the product that includes functionality, quality, complexity, efficiency, reliability, maintainability, and many other abilities.
- Project metrics can be consolidated to create process metrics that are public to the software organization as a whole. However, if the measures are normalized then it is possible to create software metrics that enable comparison to broader organizational averages. Both size-oriented and function-oriented metrics are normalized in this manner.

6.5.1 Size-Oriented Metrics

- Size-oriented software metrics are derived by normalizing quality and productivity measures. This metric also consider the size of the software product. If any software organization keeps the simple records of software development process, then a table containing the size-oriented measures can be created. The example of such a table is shown in Fig. 6.5.1.
- This table has the list of each software development projects that are already completed over the past few years and it keeps the record of their corresponding measures for project.
- To develop metrics for a project that has the similar metrics compared to other projects, the developer can select the lines of codes as a standard normalization value.
- In fact, the size-oriented metrics are not considered as foolproof metrics and it is globally not accepted as the best metrics for software process.

Project	LOC	Effort	$(000)	Pp. doc.	Errors	Defects	People
alpha	12,100	24	168	365	134	29	3
beta	27,200	62	440	1224	321	86	5
gamma	20,200	43	314	1050	256	64	6
.	.	.	.	.	.		
.	.	.	.	.	.		
.	.	.	.	.	.		

Fig. 6.5.1 : Size-oriented metrics

6.5.2 Function-Oriented Metrics

- The second important metric is function-oriented software metrics that use an amount of the work delivered by the application as a standard normalization value.
- The most commonly used function-oriented metric is Function Point (FP). The computation of the function point depends on the information domain and complexity of the software project.
- FP is function oriented software metric that is programming language independent It used in various applications that are using conventional and nonprocedural languages. It depends on the data known in the beginning of the project development or early evolution of the project. These are the reasons why FP is a good and attractive estimation approach.
- The computation of FP depends on subjective data rather than objective data. Later on, it may be difficult to keep the counts of the information domain and FP will be a simple number and it would have no physical meaning.

6.5.3 Reconciling LOC and FP Metrics

- The Lines of Code (LOC) and Function Points and their relationship are based on the programming language used to implement the software. The relationship also depends on the quality of the design. Various researches have been observed in the past that tried to relate LOC and FP measures.
- Following Table 6.5.1 provides a rough estimates of LOC required to develop a FP in different programming languages:

Table 6.5.1 : LOC per function point

Programming language	Average	Median	Low	High
Access	35	38	15	47
ASP	62	-	32	127
Assembler	337	315	91	694
C	162	109	33	704
C++	66	53	29	178
COBOL	77	77	14	400
FORTRAN	-	-	-	-
FoxPro	32	35	25	35
Informix	42	31	24	57
Java	63	53	77	-
JavaScript	58	63	42	75

Programming language	Average	Median	Low	High
JSP	59	-	-	-
Lotus Notes	21	22	15	25
Oracle	30	35	4	217
PeopleSoft	33	32	30	40
Perl	60	-	-	-
Power builder	32	31	11	105
Smalltalk	26	19	10	55
SQL	40	37	7	110
Visual Basic	47	42	16	158

- A review of these data indicates that one LOC of C++ provides approximately 2.4 times the functionality (on average) as one LOC of C.
- Practically the FP and LOC based metrics are considered as accurate measure of software development effort and cost.

6.5.4 Object-Oriented Metrics

- The above discussed conventional software project metrics (i.e. LOC or FP) are also used to estimate the object-oriented software projects. The problem with these approaches is that they do not provide schedule, effort and adjustments required.
- Following are the sets of object-oriented metrics for the object-oriented projects suggested :
  - **Number of scenario scripts** : The scenario script is actually a detailed sequence of steps used in interaction between the end-user and the product.
  - **Number of key classes :** The key classes are highly independent components defined in object-oriented analysis. These key classes sit in the centre of the problem domain.
  - **Number of support classes :** The support classes are used in system implementation but they are not directly involved in the problem domain. The examples of support classes are : User interface classes, database access classes and database manipulation classes.
  - **Average number of support classes per key class :** Normally the key classes are known in the beginning of the project itself. The support classes are defined throughout the development process. If the average number of support classes per key class is known in a problem domain then the estimation would be much simpler.
  - **Number of subsystems :** Actually a subsystem is an aggregation of keys class and its support class to define a function that is visible to the user. Once these subsystems are accurately identified, then it would be very easy to make a schedule and the related work among project staff.

6.5.5 Integrating Metrics within the Software Process

- It is a survey that most of the software developers do not go for software measurement. Here we present certain important points that will depict the significance of software measurement.
- If developer do not measure, then it becomes too difficult to find out whether there is an improvement or not. Actually there is no any way to find out the improvement except software measurement. If there is no improvement, then the progress of the project is lost.
- During the development phase of a project and at technical level, software metrics provides lot of benefits.
- Once the development is completed then developer would like to find out answers for various questions like :
  - Which of the requirements may change ?

- How much testing should be done for each component ?
- Which of the components are prone to errors ?

- If a developer has done software measurement in the beginning of development process, then it is very easy to find out the answers for all of these questions.
- Thus we can say that software metrics acts as a technical guide for development process. It is always better to integrate metrics within the software process.
- Also by **establishing a baseline** for metrics, the immediate benefits can be obtained from process, project, products i.e. at technical levels.
- The **metrics baseline** consists of the history or the past data of various software development projects as shown in Fig. 6.5.1 (Size-oriented metrics).
- The metric baseline must have following attributes for effective process improvements :
  - All the data must be accurate and there should not be any blind guesses.
  - If possible, collect the data from various projects so that comparison becomes easy.
  - All the measures must be consistent.
  - The application under development should be similar to application estimated.
- The process of establishing the metric baseline is exhibited in the Fig. 6.5.2.

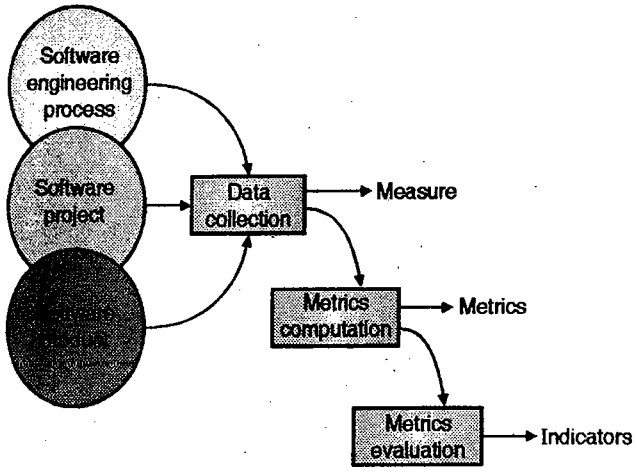


Fig. 6.5.2 : Establishing a baseline

Syllabus Topic : Effort Estimation and Scheduling

6.6 Effort Estimation and Scheduling

Review Question

Q. Explain Effort estimation and scheduling.

- The software cost estimation and the efforts estimation can never be the exact. Because there various factors affecting the estimation.

  For example : Manpower, development environment, political environment etc. These factors will affect the overall cost of the project and the efforts taken to develop the project.
- Following are some options available for achieving reliable software project estimation :
  - Try to delay the estimation until the late in the project
  - Use baseline metrics theory i.e. collect accurate estimates of past similar projects.
  - Use simple decomposition techniques to obtain cost and effort estimates.
  - Use more than one estimation models.
- Software project estimation is a form of problem solving, and in most cases, the

problem to be solved is too complex. For this reason, we decompose the problem, re-characterizing it as a set of smaller problems.

6.6.1 Software Sizing

- The accuracy of a software project estimate is predicated on a number of things :
  - The degree estimation for the size of the software to be developed.
  - The ability to translate the estimation for the size into the human efforts, time, and money.
  - The abilities of the development team reflected in the project plan, and
  - The product stability requirements and the environment that supports the software development efforts.
- Four different approaches to the sizing problem have been suggested :
  - **Fuzzy logic sizing :** In order to apply the fuzzy logic approach, the planner should identify the type of software and should establish its magnitude on a qualitative scale. Later on, this magnitude is refined within the original range.
  - **Function point sizing :** In this, the planner establishes the estimates of the information domain characteristics.
  - **Standard component sizing :** In standard component sizing approach, the software consist of different standard components. Examples of the standard components for an information system are :
    1. Subsystems
    2. Modules
    3. Screens
    4. Reports
    5. Interactive programs,
    6. Batch programs, files LOC
    7. Object-level instructions
  - The project planner should estimate the number of occurrences of each of the standard components.
  - **Change sizing :** In this approach, the planner should estimate different modifications that is accomplished . This particular approach is generally used when a project uses some existing project and requires its modification.
- Results of each of these sizing approaches should be combined statistically to create a three-point or expected-value estimate.

6.6.2 Problem-Based Estimation

- The LOC and FP data are generally used during software project estimation in following two ways required :
  (1) As an estimation variable that will enable the sizing of each element of the software and
  (2) Secondly, as a baseline metrics used to develop cost and effort estimates.
- The project planner starts with a software scope that is a bounded statement, and tries to decompose the software into problem functions. These problem functions can be estimated individually. Later the LOC or FP is estimated for each function.
- Following are baseline productivity metrics :
  - LOC/person-month and
  - FP/person- month
- These baseline productivity metrics are applied to the appropriate estimation variable and from it the cost or effort for the function is derived. Finally the function estimates are integrated to produce one complete estimate for the whole project.
- The expected-value or three-pint value can be computed for the estimation variable

size S, as a weighted average of the optimistic size (S_{opt}), most likely size (S_m), and pessimistic size (S_{pess}) estimates. Consider the following example,

$$S = (S_{opt} + 4S_m + S_{pess}) / 6 \quad ...(1)$$

- The above example gives the emphasis on "most likely" estimate and it follows a beta probability distribution.

6.6.3 An Example of LOC-Based Estimation

As an example of LOC and FP problem-based estimation techniques, consider a software package to be developed for a computer-aided design application for mechanical components. The software is to execute on an engineering workstation and must interface with various peripherals including a mouse, digitizer, high-resolution colour display, and laser printer. A preliminary statement of software scope can be developed.

Table 6.6.1 : Estimation Tables for the LOC Methods

Function	Estimated LOC
User Interface and Control Facilities (UICF)	2,300
Two-Dimensional Geometric Analysis (2DGA)	5,300
Three-Dimensional Geometric Analysis (3DGA)	6,800
Database Management (DBM)	3,350
Computer Graphics Display Facilities (CGDF)	4,950
Peripheral Control Function (PCF)	2,100
Design Analysis Modules (DAM)	8,400
Estimated lines of code	**33,200**

For our purposes, we assume that further refinement has occurred and that the major software functions listed in Table 6.6.1 are identified. Following the decomposition technique for LOC, an estimation table, shown in Table 6.6.1, is developed. A range of LOC estimates is developed for each function. **For example :** The range of LOC estimates for the 3D geometric analysis function is optimistic 4600 LOC, most likely 6900 LOC, and pessimistic 8600 LOC. Applying Equation (1), the expected value for the 3D geometric analysis function is 6800 LOC.

6.6.4 An Example of FP-Based Estimation

Table 6.6.2 : Estimating information domain values

Information domain value	Opt.	Likely	Pess.	Est. count	Weight	FP count
Number of external inputs	20	24	30	24	4	97
Number of external outputs	12	15	22	16	5	78
Number of external inquiries	16	22	28	22	5	88
Number of internal logical files	4	4	5	4	10	42
Number of external interface files	2	2	3	2	7	15
Count total						320

Decomposition for FP-based estimation focuses on information domain values rather than software functions. Referring to the table presented in Table 6.6.2, the project planner estimates external inputs, external outputs, external inquiries, internal logical files, and external interface files for the CAD software. FP are computed using the technique discussed For the purposes of this estimate, the complexity weighting factor is assumed to be average. Table 6.6.2 presents the results of this estimate.

Each of the complexity weighting factors is estimated and the value adjustment factor is computed.

	Factor	Value
1.	Backup and recovery	4
2.	Data communications	2
3.	Distributed processing	0
4.	Performance critical	4
5.	Existing operating environment	3

	Factor	Value
6.	On-line data entry	4
7.	Input transaction over multiple screens	5
8.	ILFs updated online	3
9.	Information domain values complex	5
10.	Internal processing complex	5
11.	Code designed for reuse	4
12.	Conversion/installation in design	3
13.	Multiple installations	5
14.	Application designed for change	5
	Value adjustment factor	**1.17**

Finally, the estimated number of FP is derived:

$FP_{estimated}$ = count – total × [0.65 + 0.01 $\sum(F_i)$]

$FP_{estimated}$ = 375

The organizational average productivity for systems of this type is 6.5 FP/pm.

6.6.5 Process-Based Estimation

- The most common technique for estimating a project is to base the estimate on the process that will be used.
- That is, the process is decomposed into a relatively small set of tasks and the effort required to accomplish each task is estimated.
- A series of framework activities must be performed for each function. Functions and related framework activities may be represented as part of a table similar to the one presented in Table 6.6.3.
- After the problem functions and development process activities are declared, the planner may begin estimating the effort like person-months.
- This estimation is necessary to accomplish each of the software process activity for each of software function. All these data constructs the central matrix of the table as shown in Table 6.6.3.
- The average labour rates like cost/unit effort are applied to the effort estimated for each of the process activity.

Table 6.6.3 : Process-based estimation table

Activity →	CC	Planning	Risk analysis	Engineering		Construction release		CE	Totals
Task →				Analysis	Design	Code	Test		
Function									
↓									
UICF				0.50	2.50	0.40	5.00	n/a	8.40
2 DGA				0.75	4.00	0.60	2.00	n/a	7.35
3DGA				0.50	4.00	1.00	3.00	n/a	8.50
CGDF				0.50	3.00	1.00	1.50	n/a	6.00
DBM				0.50	3.00	0.75	1.50	n/a	5.75
PCF				0.25	2.00	0.50	1.50	n/a	4.25
DAM				0.50	2.00	0.50	2.00	n/a	5.00
Totals	**0.25**	**0.25**	**0.25**	**3.50**	**20.50**	**4.50**	**16.50**		**46.00**
% effort	**1%**	**1%**	**1%**	**8%**	**45%**	**10%**	**36%**		

CC = Customer Communication, CE = Customer Evaluation

6.6.6 An Example of Process-Based Estimation

- To explain the use of process-based estimation, we again consider the CAD software. Refer the completed process-based table shown in Table 6.6.3, estimates of effort (in person-months) for each software engineering activity are provided for each CAD software function. The engineering and construction release activities are subdivided into the major software engineering tasks.
- Generally the gross estimates of effort are prepared for communication with the customers, planning and risk analysis. It must be noted that nearly 53 percent of the efforts is spent on front-end engineering tasks like requirements analysis and design. It actually indicates the significance and importance of this work.
- **For example :** Let the average burdened labour rate of $8,000 per month, then the total estimated project cost is computed as $368,000, and the estimated effort is approximately 46 person-months. In this case, the labour rates could be associated with each framework activity, if desired. It can also be associated with each of the software engineering task and calculated separately.

6.6.7 Estimation with Use-Cases

- Use-cases give software team members to see into insight of the software scope and requirements. But use-cases estimation approach can be problematic due to following reasons :
  - Use-cases may be written by using various formats and styles. So far, there is no standard format.
  - They represent the external view of the software and may be described at various levels of abstraction.
  - They do not look at the complexity of the functions and features.
  - And use-cases do not explain complex behaviour also.
  - In contrast to a LOC or a FP computations, the efforts required to implement one person's "use-case" and some other person's "use-case "may may vary drastically. That is, one person may require months and another person may require a day or two.
- To illustrate how this computation might be made, consider the following relationship :

$$LOC_{estimate} = N \times LOC_{avg} + [(S_a/S_h - 1) + (P_a/P_h - 1)] \times LOC_{adjust} \quad ...(2)$$

where, N = Actual number of use-cases

LOC_{avg} = Historical average LOC per use-case for this type of subsystem.

LOC_{adjust} = Represents an adjustment based on of LOC_{avg}

S_a = Actual scenarios per use-case

S_h = Average scenarios per use-case for this type of subsystem

P_a = Actual pages per use-case

P_h = Average pages per use-case for this type of subsystem

- The Equation (6.6.2) is useful in developing the rough estimate of the number based on LOC and the actual number of use-cases used by several scenarios along with the page length of the use-cases. Thus the adjustment shows the historical average LOC per use case.

6.6.8 An Example of Use-Case Based Estimation

- The CAD software is composed of three subsystem groups :

  1. User interface subsystem
  2. Engineering subsystem group
  3. Infrastructure subsystem group

- Consider an example in that six use-cases describe the **user interface subsystem**. Each of the use case is written by approximately 10 scenarios and it has an average length of six pages.
- The **engineering subsystem group** is illustrated by approximately 10 use-cases. And each of these use-cases may have 20 scenarios associated with it and it has an average length of nearly eight pages.
- Finally, the **infrastructure subsystem group** is covered by five use-cases with an average of only six scenarios and an average length of five pages.
- Using the relationship noted in Equation (2), the table shown in Table 6.6.4 is developed.
- Hence the LOC estimate for the user interface subsystem is computed using Equation (2). Using the same approach, estimates are made for both the engineering and infrastructure subsystem groups. Table 6.6.4 summarizes the estimates and indicates that the overall size of the CAD software is estimated at 42,500 LOC.

6.6.9 Reconciling Estimates

- The estimation techniques discussed in the preceding sections result in multiple estimates which must be reconciled to produce a single estimate of effort, project duration, or cost. To illustrate this reconciliation procedure, we again consider the CAD software.
- Total estimated effort for the CAD software range from a low of 46 person-months (derived using a process-based estimation approach) to a high of 68 person-months (derived with use-case estimation). The average estimate is 56 person-months.
- The variation from the average estimate is approximately 18 percent on the low side and 21 percent on the high side. The planner must determine the cause of divergence and then reconcile the estimates.

Table 6.6.4 : Use-case estimation

	Use-cases	Scenarios	Pages	Scenarios	Pages	LOC	LOC estimate
User interface subsystem	6	10	6	12	5	560	3,366
Engineering subsystem group	10	20	8	16	8	3100	31,233
Infrastructure subsystem group	5	6	5	10	6	1650	7,970
Total LOC estimate							**42,568**

6.7 Empirical Estimation Models

Review Question

Q. Explain Empirical Estimation Models.

- An estimation model for software products uses empirically derived formulas in order to predict the effort in terms of functions of LOC or FP. The values for LOC or FP can be estimated. Here, the resultant values for LOC or FP are used in the estimation model without the values using the tables.
- The empirical data are always supported in most of the estimation models. These empirical data are derived from some sample of projects. Due to this reason only, there is no estimation model suitable for all the classes of software and in all development environments. And hence the results obtained in such cases are used judiciously.
- An estimation model must be calibrated in such a way that reflects its local conditions. The estimation model must be thoroughly tested by applying the data gathered from the past completed projects.

Syllabus Topic : Project scheduling

6.8 Project Scheduling

Review Question

Q. Explain Project Scheduling.

6.8.1 The Structure of Estimation Models

- The estimation model is generally derived from regression analysis on the data that are collected from previously collected software projects. The structure of estimation models take the following form :

$$E = A + B \times (e_v)^c \quad ...(1)$$

Where A, B, and C are nothing but the empirically derived constants. The constant E is an effort in person-months, and the estimation variable is e_v.

- In addition to the relationship described by Equation (1), most of estimation models use project adjustment component that enables the effort E. Generally the effort E is adjusted by the following project characteristics :
  - o Problem complexity,
  - o Staff experience,
  - o Development environment etc.
- Various LOC-oriented estimation models that have been proposed in the literature are as follows :

1) $E = 5.288 \times (KLOC)^{1.047}$

(Doty model for KLOC > 9)

2) $E = 3.2 \times (KLOC)^{1.05}$

(Boehm simple model)

3) $E = 5.2 \times (KLOC)^{0.91}$

(Walston-Felix model)

4) $E = 5.5 + 0.73 \times (KLOC)^{1.16}$

(Bailey-Basili model)

- Following are some FP-oriented models proposed :

1) $E = -37 + 0.96$ FP (Kemerer model)

2) $E = -91.4 + 0.355$ FP

(Albrecht and Gaffney model)

3) $E = -12.88 + 0.405$ FP

(small project regression model)

- After observing all these models, we come to conclusion that different results are possible for the same values of LOC or FP.

6.8.2 The COCOMO II Model SPPU - May 14

University Question

Q. Explain the COCOMO-II estimation model. **(May 2014, 5 Marks)**

- **Barry Boehm** has devised a software estimation models hierarchy having the name as COCOMO i.e. **CO**nstructive **CO**st **MO**del. The COCOMO model has been

evolved into more comprehensive model called COCOMO II. It is actually having a hierarchy of estimation models that focus the following areas :

- **Application composition model :** It is used in the early stages of development, when user interfaces, system interaction, performance assessment and technology evaluation are prime important.
- **Early design stage model :** Once the requirements are stabilized and basic architecture is constructed, then early design stage model is used.
- **Post-architecture stage model :** It is used during development of the software.

- COCOMOII models also require sizing information like other estimation models for the software. Following three sizing options are available :
  - Object points
  - Function points and
  - Lines of source code
- The COCOMO II model uses object points that are an indirect software measure. They are is computed using the counts of :
  - Screenshots taken at user interface
  - Reports and
  - Components required in developing the application.
- The screenshots or the reports are classified into any of the following three complexity levels :
  - Simple
  - Medium
  - Difficult
- The client and server data tables are required to create the screenshots and reports. The complexity is defined as the function of these reports.

Table 6.8.1 : Complexity weighting for object types

Object type	Complexity weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL component			10

- After the determination of complexity, the number of screenshots, reports, and components object point are weighted as per the table shown in Table 6.8.1. In component-based development when software reuse is implemented, then the percent of reuse (% reuse) is estimated and the object point count is adjusted according to the following equation :

NOP = (object points) × [(100 – % reuse)/100]

where NOP → New Object Points.

Table 6.8.2 : Productivity rate for object points

Developer's experience/capability	Very low	Low	Nominal	High	Very high
Environment maturity / capability	Very low	Low	Nominal	High	Very high
PROD	4	7	13	25	50

- In order to derive the estimate of effort, a "productivity rate" should be derived first. They are based on the computed NOP value. The Table 6.8.2 shown above presents the productivity rate. These productivity rates are illustrated for various levels of developer's experience as well as various levels of environment maturity.

PROD = NOP/person-month

- After determining the productivity rate, an estimate of project effort can be derived as follows :

Estimated effort = NOP/PROD

6.8.3 The Software Equation

- The software equation is a model that considers a specific distribution of effort over the project development life. The software equation model is a multivariable model. This model is derived from productivity data collected from nearly 4000 contemporary software projects. Depending on all these data, the estimation model takes the following form :

$$E = [LOC \times B^{0.333}/P]^3 \times (1/t^4) \quad ...(2)$$

Where,

E = Person-months effort or person-years efforts

t = Duration of the project in months or years

B = The special skills factor

P = Productivity parameter

- The productivity parameter reflects the following characteristics :
  - Process maturity
  - Management practices
  - Good software engineering practices
  - Programming languages used
  - Software environment
  - The skill set of team members
  - The experience of team members, and
  - The complexity of the software
- Typical values might be P = 2000 for development of real-time embedded software;
- P = 10,000 for telecommunication and systems software; P = 28,000 for business systems applications. The productivity parameter is derived using historical data collected from previous development efforts.
- The software equation has following two independent parameters :
  - An estimate of size and
  - An indication of project duration.

6.9 Importance of Project Schedules

Review Question

Q. What is the Importance of Project Schedules ?

- To organize and complete your projects in a timely, quality and financially responsible manner, you need to schedule projects carefully.
- Effective project scheduling plays a crucial role in ensuring project success. To keep projects on track, set realistic time frames, assign resources appropriately and manage quality to decrease product errors.
- This typically results in reduced costs and increased customer satisfaction. Important factors include financial, documentation, management and quality assurance.

Financial

- Project scheduling impacts the overall finances of a project. Time constraints require project managers to schedule resources effectively. This is particularly true when resources must have highly specialized skills and knowledge in order to complete a task or when costly materials are required.
- Completing a project in a short time frame typically costs more because additional resources or expedited materials are needed. With accurate project scheduling, realistic estimates and accurate projections prevent last-minute orders that drive up costs.

Documentation

- Creating a comprehensive work breakdown structure allows you to create a chart, such as a Gantt chart, that lists the project tasks, shows dependencies and defines milestones. Management consultant Henry Gantt designed this type of chart to show a graphic schedule of planned work. Its role in business projects is to record and report progress toward project completion.
- Your project schedule also allows you to assign human resources to the work and

evaluate their allocation to ensure you have the appropriate levels of utilization. You may also develop a program evaluation and review technique chart, or PERT chart, to help you analyze project tasks.

Management

- Effective project managers conduct regular meetings to get status reports. They use project scheduling meetings to check in with their team members and prevent costly misunderstandings.
- These regular meetings ensure that work flows from one process to the next and that each team member knows that he needs to do to contribute the project's overall success.

Quality

- Project scheduling ensures one task gets completed in a quality manner before the next task in the process begins. By assuring that quality measures meet expectations at every step of the way, you ensure that managers and team members address problems as they arise and don't wait until the end.
- No major issues should appear upon completion because you've established quality controls from the very beginning of the scheduling process.
- Effective project managers understand that ensuring quality control involves managing risks and exploiting opportunities to speed up the schedule when possible to beat the competition and achieve or maintain a competitive edge with a more reliable product.

Syllabus Topic : Estimating Activity Resources

6.10 Estimating Activity Resources

Review Question

Q. What is Estimating Activity Resources ?

- Estimate Activity Resources is the process of estimating the type and quantities of material, human resources, equipment, or supplies required to perform each activity.
- The key benefit of this process is that it identifies the type, quantity, and characteristics of resources required to complete the activity which allows more accurate cost and duration estimates. The inputs, tools and techniques, and outputs of this process are depicted in the following Fig. 6.10.1.

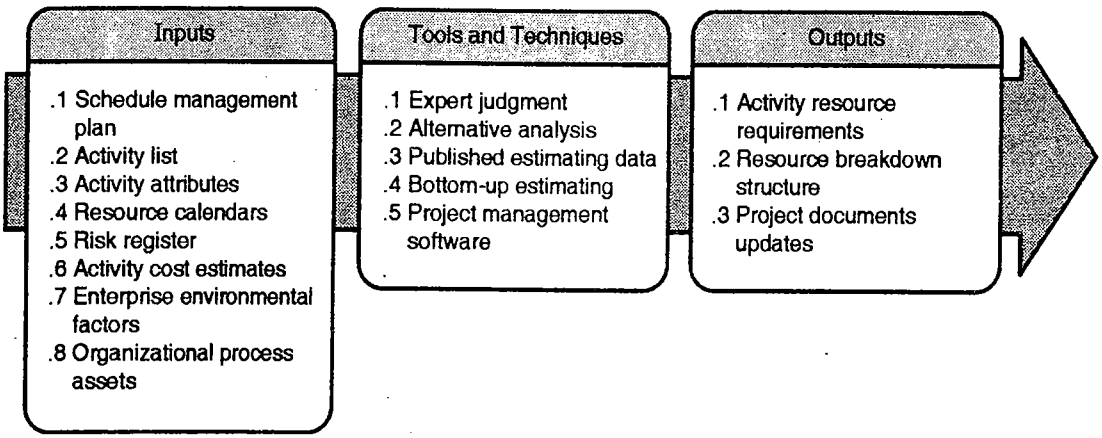


Fig. 6.10.1 : Estimating Activity Resources : Inputs, Tools & Techniques, and Outputs

- Fig. 6.10.2 depicts the data flow diagram of the process.

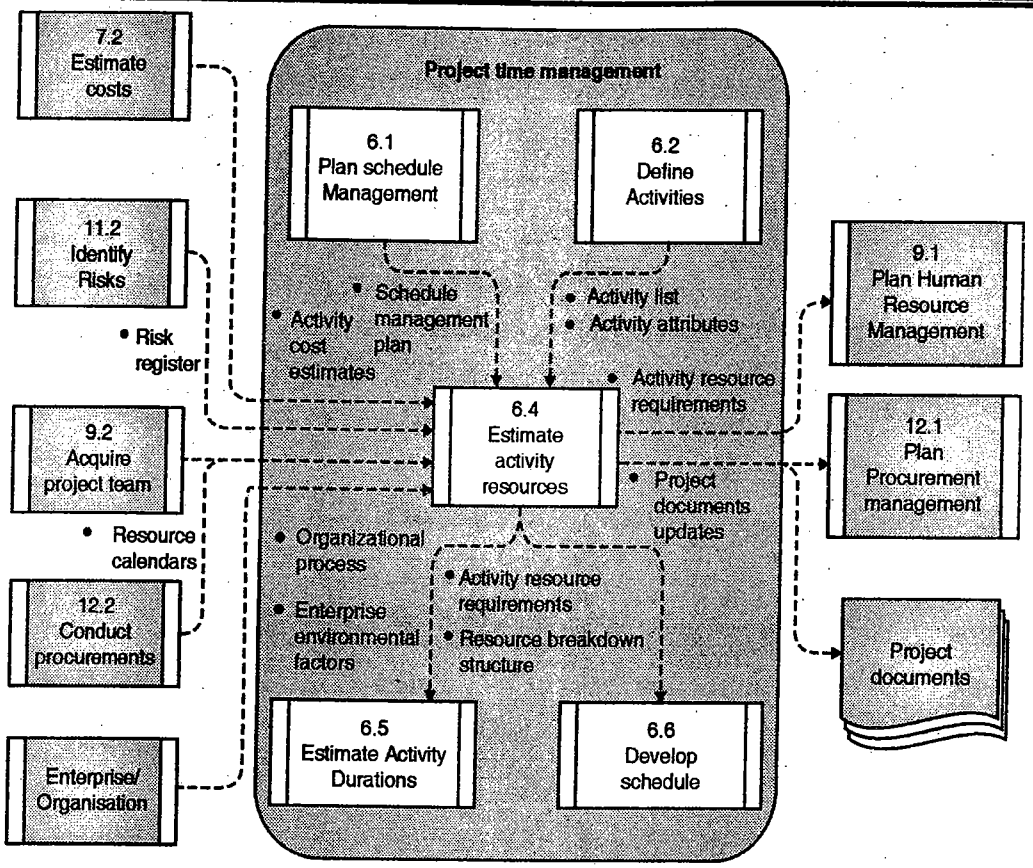


Fig. 6.10.2 : Estimating Activity Resources : Data Flow Diagram

Syllabus Topic : Estimating Activity Durations

6.11 Estimating Activity Durations

Review Question

Q. What is Estimating Activity Durations ?

- Estimate Activity Durations is the process of estimating the number of work periods needed to complete individual activities with estimated resources.
- The key benefit of this process is that it provides the amount of time each activity will take to complete, which is a major input into the Develop Schedule process. The inputs, tools and techniques, and outputs of this process are depicted in Fig. 6.11.1

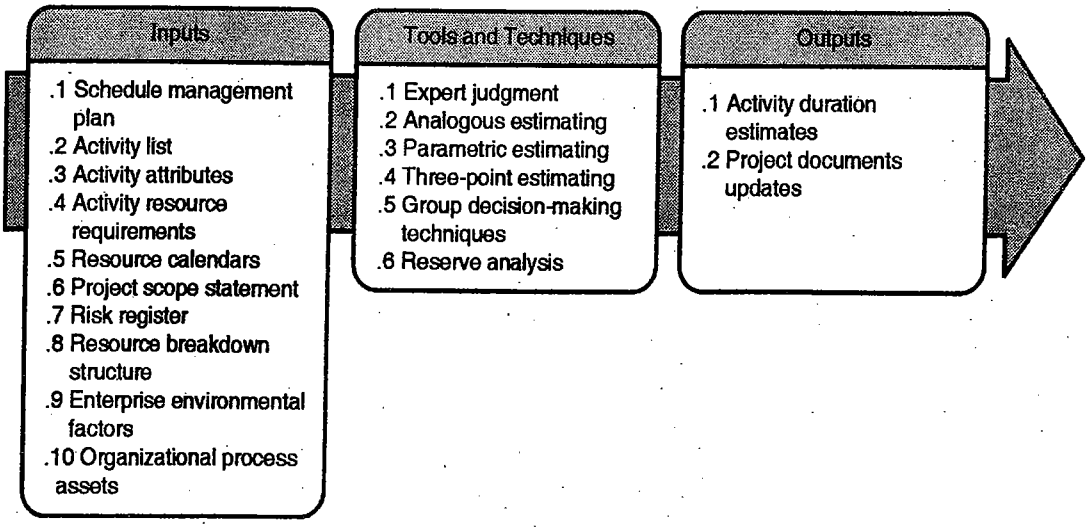


Fig. 6.11.1 : Estimating Activity Durations : Inputs, Tools & Techniques, and Outputs

- Fig. 6.11.2 depicts the data flow diagram of the process.

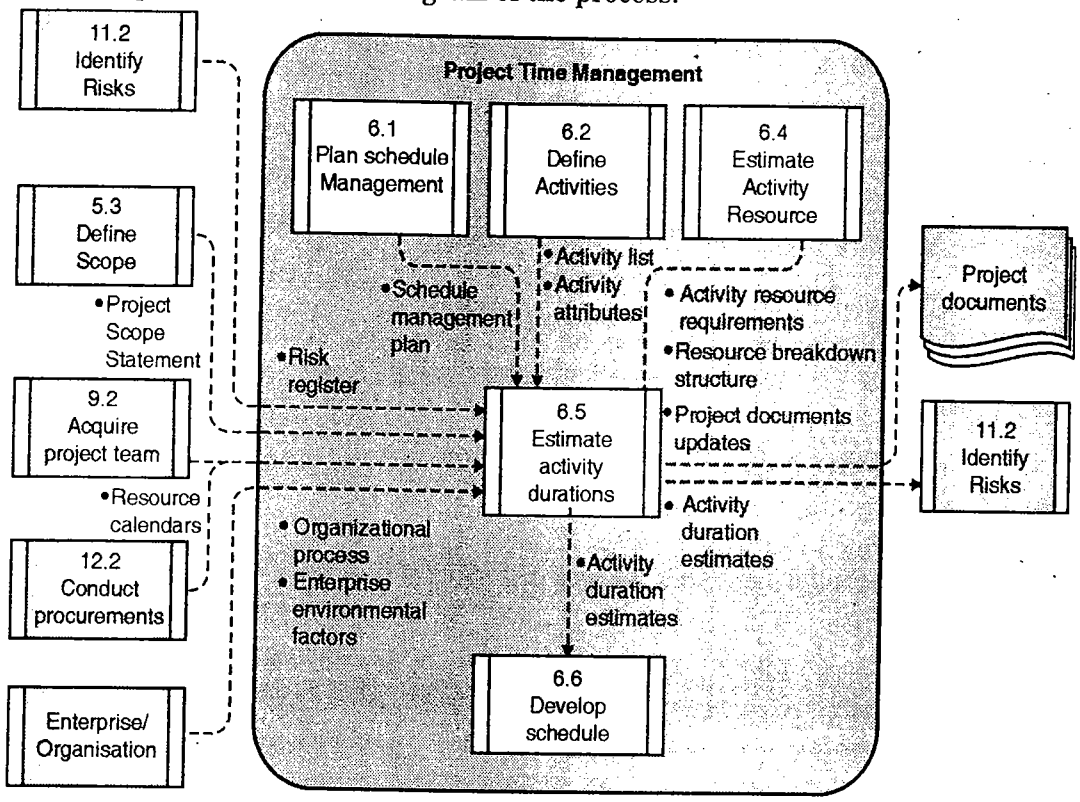


Fig. 6.11.2 : Estimating Activity Durations : Data Flow Diagram

Syllabus Topic : Developing the Schedule using Gantt Charts

6.12 Developing the Schedule using Gantt Charts

Review Question

Q. How Schedules are developed using Gantt Charts ?

- A time-line chart, also called Gantt chart, is generated on the basis of the start date inputs for each task.
- Fig. 6.12.1 shows an example Gantt chart. It elaborates the project schedule.
- In the left hand column of the Fig. 6.12.1, all the project tasks are listed. The horizontal lines indicate the duration of the task.
- For concurrent activities, multiple horizontal lines are used.

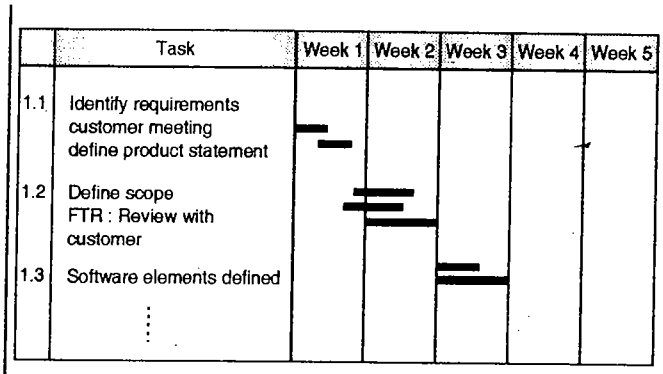


Fig. 6.12.1 : An example time-line chart

6.12.1 Tracking the Schedule

- The project schedule is nothing but a road map that defines the tasks and the milestones to be tracked.
- Tracking is achieved in different ways as follows :
  - o Conduct periodic meeting to find out the progress status and problems.
  - o Evaluate the process.

- Check whether all the activities are completed within the deadline or by the schedule date.
- Compare the planned date and actual start date for each activity.
- Meet practitioners informally to assess the progress.

- All these tracking techniques are used by the project managers.

6.12.2 Schedule and Cost Slippage

- A schedule slippage in project planning is described as missing a deadline and project is not completed in the estimated time. In order to avoid schedule slippage, the project manager must plan the project very carefully so that there should not delays in the schedule.
- The time-line charts or Gantt charts are very useful tools to plan the project in an effective manner.
- If there is slippage in schedule, then ultimately there is slippage in the cost of the project or the budget of the project.
- If the schedule slippage is detected in some projects, then project starts floating towards the upper limits of project deadline. The float in a schedule means delays and leads time mentioned in the project schedule. It is the responsibility of a project manager to diagnose the problems and find the root causes of the slippage and take necessary actions to correct the problems.
- Practically it is very difficult to achieve the deadline estimated. Following are some important tasks to be accomplished in order to avoid schedule slippage and cost slippage :
  - Identify the basic reasons behind slippage.
  - Forecast a probable delivery date.
  - Acknowledge the problems of slippage and find suitable remedy for each of the problems.
  - Try to use some method to neutralize the slippage.
  - Always discuss customers with the current progress of the project.
- **Causes of schedule and cost slippage :**

  Following are three major causes of schedule and cost slippage :
  - Poor estimation of schedule and cost.
  - Poor requirement analysis and requirement gathering, and
  - Management issues

6.13 Earned Value Analysis (EVA)

SPPU - May 12, May 13

University Question

Q. Write short on: Earned value analysis. **(May 2012, May 2013, 6 Marks)**

Review Question

Q. Explain Earned Value Analysis (EVA).

- During the project tracking, each developer gives his project or task progress status to manager. But this assessment of information is subjective.
- There should be some quantitative technique for assessing progress of project.
- Earned Value Analysis (EVA) is a technique used for quantitative analysis in the following manner :
- The EVA provides a common value scale for every task. The time to complete total project is estimated and every task is given an earned value based on its estimated percentage of the total.
- To determine the earned value, following steps are performed :
  1. The Budgeted Cost of Work Scheduled (BCWS) is calculated for each task.

2. The BCWS values for all work tasks are summed to calculate the final budget.

∴ BAC = Σ(BCWSk) for all tasks k.

Where BAC = Budget at Completion.

3. Now, the value for Budgeted Cost of Work Performed (BCWP) is computed. The value of BCWP is sum of the BCWS values for all work tasks completed by that point in time.

- The progress indicators can be calculated as :
  - Scheduled Performance Index,

$$SPI = \frac{BCWP}{BCWS}$$

  - Scheduled Variance,

$$SV = BCWP - BCWS$$

- An SPI value close to 1.0 indicates efficient execution of the project schedule.

Percent scheduled for completion

$$= \frac{BCWS}{BAC}$$

- It provides the percentage of work that should have been completed by time t.

$$\text{Percent complete} = \frac{BCWP}{BAC}$$

- It provides the percent of completeness of the project at a given point in time t.
- Also, the Actual Cost of Work Performed (ACWP) is the sum of the efforts actually expanded on work tasks that have been completed by a point in time on the project schedule.

∴ Cost performance index,

$$CPI = \frac{BCWP}{ACWP}$$

Cost variance, CV = BCWP – ACWP

- A CPI value close to 1.0 provides a strong indication that the project is within its defined budget.

6.14 Project Scheduling Tools and Techniques

Review Question

Q. What are different Project Scheduling Tools and Techniques ?

- Scheduling of different engineering activities can use the available project scheduling tools and techniques.
- Following are some project scheduling tools and techniques :
  - CPM (Critical Path Method)
  - PERT (Program Evaluation and Review Technique)
- These are two project scheduling methods that can be applied to the software development process.
- Both tools use the data and information received from the earlier developments.
- Both tools allow to determine the critical path, duration of the projects, and time estimate for the individual activity, efforts taken, duration.

6.14.1 CPM (Critical Path Method)

- The critical path method (CPM) is a project modeling technique developed in the late 1950s. CPM is commonly used with all forms of projects, including construction, aerospace and defence, software development, research projects, product development, engineering, and plant maintenance, among others.
- Any project with interdependent activities can apply this method of mathematical analysis. The first time CPM was used for major skyscraper development was in 1966 while constructing the former World Trade Centre Twin Towers in NYC.
- Although the original CPM program and approach is no longer used, the term is generally applied to any approach used to analyze a project network logic diagram.

- The essential technique for using CPM is to construct a model of the project that includes the following :
  1. A list of all activities required to complete the project
  2. The time (duration) that each activity will take to complete,
  3. The dependencies between the activities and,
  4. Logical end points such as milestones or deliverable items.
- Using these values, CPM calculates the longest path of planned activities to logical end points or to the end of the project, and the earliest and latest that each activity can start and finish without making the project longer.
- This process determines which activities are "critical" (i.e., on the longest path) and which have "total float" (i.e., can be delayed without making the project longer).
- In project management, a critical path is the sequence of project network activities which add up to the longest overall duration, regardless if that longest duration has float or not. This determines the shortest time possible to complete the project.
- A project can have several, parallel, near critical paths; and some or all of the tasks could have 'free float' and/or 'total float'. An additional parallel path through the network with the total durations shorter than the critical path is called a sub-critical or non-critical path. Activities on sub-critical paths have no drag, as they are not extending the project's duration.
- CPM analysis tools allow a user to select a logical end point in a project and quickly identify its longest series of dependent activities (its longest path). These tools can display the critical path (and near critical path activities if desired) as a cascading waterfall that flows from the project's start (or current status date) to the selected logical end point.

6.14.2 PERT (Program Evaluation and Review Technique)

- The program (or project) evaluation and review technique, commonly abbreviated PERT, is a statistical tool, used in project management, which was designed to analyze and represent the tasks involved in completing a given project.
- It is commonly used in conjunction with the critical path method (CPM).
- PERT is a method of analyzing the tasks involved in completing a given project, especially the time needed to complete each task, and to identify the minimum time needed to complete the total project.
- PERT was developed primarily to simplify the planning and scheduling of large and complex projects. It was developed for the U.S. Navy Special Projects Office in 1957 to support the U.S. Navy's Polaris nuclear submarine project.
- It was able to incorporate uncertainty by making it possible to schedule a project while not knowing precisely the details and durations of all the activities.
- It is more of an event-oriented technique rather than start- and completion-oriented, and is used more in projects where time is the major factor rather than cost.
- It is applied to very large-scale, one-time, complex, non-routine infrastructure and Research and Development projects.
- The first step to scheduling the project is to determine the tasks that the project requires and the order in which they must be completed. The order may be easy to record for some tasks (e.g. When building a house, the land must be graded before the foundation can be laid) while difficult for others (there are two areas that need to be

graded, but there are only enough bulldozers to do one).

- Additionally, the time estimates usually reflect the normal, non-rushed time. Many times, the time required to execute the task can be reduced for an additional cost or a reduction in the quality.
- PERT planning involves the following steps :

1. Identify the specific activities and milestones.
2. Determine the proper sequence of the activities.
3. Construct a network diagram.
4. Estimate the time required for each activity.
5. Determine the critical path.
6. Update the PERT chart as the project progresses

1. Identify the specific activities and milestones

- The activities are the tasks required to complete a project. The milestones are the events marking the beginning and the end of one or more activities.
- It is helpful to list the tasks in a table that in later steps can be expanded to include information on sequence and duration.

2. Determine the proper sequence of the activities

- This step may be combined with the activity identification step since the activity sequence is evident for some tasks.
- Other tasks may require more analysis to determine the exact order in which they must be performed.

3. Construct a network diagram

- Using the activity sequence information, a network diagram can be drawn showing the sequence of the serial and parallel activities.
- Each activity represents a node in the network, and the arrows represent the relation between activities.
- Software packages simplify this step by automatically converting tabular activity information into a network diagram.

4. Estimate the time required for each activity

- Weeks are a commonly used unit of time for activity completion, but any consistent unit of time can be used.
- A distinguishing feature of PERT is its ability to deal with uncertainty in activity completion time. For each activity, the model usually includes three time estimates:
  - (i) **Optimistic time** – generally the shortest time in which the activity can be completed. It is common practice to specify optimistic time to be three standards deviations from the mean so that there is a approximately a 1% chance that the activity will be completed within the optimistic time.
  - (ii) **Most likely time** – the completion time having the highest probability. Note that this time is different from the expected time.
  - (iii) **Pessimistic time** – the longest time that an activity might require. Three standard deviations from the mean is commonly used for the pessimistic time.
- PERT assumes a beta probability distribution for the time estimates.
- For a beta distribution, the expected time for each activity can be approximated using the following weighted average :

Expected time = (Optimistic + 4 x Most likely + Pessimistic) / 6

- This expected time may be displayed on the network diagram.
- To calculate the variance for each activity completion time, if three standard deviation times were selected for the optimistic and pessimistic times, then there are six standard deviations between them, so the variance is given by:

[(Pessimistic - Optimistic) / 6]

5. Determine the critical path

- The critical path is determined by adding the times for the activities in each sequence and determining the longest path in the project. The critical path determines the total calendar time required for the project.
- If activities outside the critical path speed up to slow down (within limits), the total project time does not change. The amount of time that a non – critical path activity can be delayed without the project is referred to as a slack time.
- If the critical path is not immediately obvious, it may be helpful to determine the following four quantities foe each activity :

  ES - Earliest Start time

  EF - Earliest Finish time

  LS - Latest Start time

  LF - Latest Finish time
- These times are calculated using the expected time for the relevant activities. The earliest start and finish times of each activity are determined by working forward through the network and determining the earliest time at which an activity can start and finish considering its predecessors activities.
- The latest start and finish times are the latest times that an activity can start and finish without delaying the project. LS and LF are found by working backward through the network. The difference in the latest and earliest finish of each activity is that activity's slack. The critical path then is the path through the network in which none of the activities have slack.
- Since the critical path determines the completion date of the project, the project can be accelerated by adding the resources required to decrease the time for the activities in the critical path. Such a shortening of the project sometimes is referred to as **project crashing**.

6. Update the PERT chart as the project progresses

- Make adjustments in the PERT chart as the project progresses. As the project unfolds, the estimated times can be replaced with actual times.
- In cases where there are delays, additional resources may be needed to stay on schedule and the PERT chart may be modified to reflect the new situation.

6.14.2.1 Advantages using PERT

PERT is useful because it provides the following information :

- Expected project completion time;
- Probability of completion before a specified date;
- The critical path activities that directly impact the completion time;
- The activities that have slack time and that can be lend resources to critical path activities;
- Activity start and end date.

6.15 Planning Cost Management

Review Question

Q. Discuss Cost Management.

- Project cost management is traditionally a weak area of IT projects. IT project managers must acknowledge the importance of cost management and take responsibility for understanding basic cost concepts, cost estimating, budgeting, and cost control.
- Project managers must understand several basic principles of cost management to be effective in managing project cost. Important concepts include :
  - Profits and profit margins,
  - Life cycle costing,
  - Cash flow analysis,
  - Sunk cost, and
  - Learning curve theory.
- Planning cost management involves determining the policies, procedures, and documentation that will be used for planning, executing, and controlling project cost. The main output of this process is a cost management plan.
- Estimating costs is a very important part of project cost management. There are several types of cost estimates, including rough order of magnitude (ROM), budgetary, and definitive.
- Each type of estimate is done during different stages of the project life cycle, and each has a different level of accuracy. Several tool and techniques can help you develop cost estimates, including analogous estimating, bottom-up estimating, parametric estimating, and computerized tools.
- Determining the budget involves allocating costs to individual work items over time. It is important to understand how particular organizations prepare budgets so estimates are made accordingly.
- Controlling cost includes monitoring cost performance, reviewing changes, and notifying project stakeholders of changes related to cost. Many basic accounting and finance principles relate to cost project management.
- Earned value management is an important method used for measuring project performance. Earned value management integrates scope, cost, and schedule information.
- Project portfolio management allows organization to collect and control an entire suite of projects or investments as one set of interrelated activities. Several software products can assists with project cost management. Enterprise project management software and portfolio management software can help managers evaluate data on multiple projects..

6.15.1 Estimating Costs

- The software cost estimation and the efforts estimation can never be the exact. Because there various factors affecting the estimation.

  For example : Manpower, development environment, political environment etc. These factors will affect the overall cost of the project and the efforts taken to develop the project.
- Following are some options available for achieving reliable software project estimation :
  - Try to delay the estimation until the late in the project
  - Use baseline metrics theory i.e. collect accurate estimates of past similar projects.
  - Use simple decomposition techniques to obtain cost and effort estimates.
  - Use more than one estimation models.

- Software project estimation is a form of problem solving, and in most cases, the problem to be solved is too complex. For this reason, we decompose the problem, re-characterizing it as a set of smaller problems.

6.15.2 Types of Cost Estimates

- Cost estimation includes the process or methods that help us in predicting the actual and total cost that will be needed for our software and is considered as one of the complex and challenging activity for the software companies.
- Their goal is to develop software which is cheap and at the same time deliver good quality. Software cost estimation is used basically by system analysts to get an approximation of the essential resources needed by a particular software project and their schedules.
- Important parameters in estimating cost are size, time, effort etc. Process of software estimation basically focuses on four steps. At first we estimate software size, then the needed effort after this we derive the schedule and at last calculate cost of the software.
- Basically the cost estimations can be broadly categorized into following two types :
  - o Problem-Based Estimation
  - o Process-Based Estimation

6.15.2.1 Problem-Based Estimation

- The LOC and FP data are generally used during software project estimation in following two ways required :
  (1) As an estimation variable that will enable the sizing of each element of the software and
  (2) Secondly, as a baseline metrics used to develop cost and effort estimates.
- The project planner starts with a software scope that is a bounded statement, and tries to decompose the software into problem functions. These problem functions can be estimated individually. Later the LOC or FP is estimated for each function.
- Following are baseline productivity metrics :
  - o LOC/person-month and
  - o FP/person- month
- These baseline productivity metrics are applied to the appropriate estimation variable and from it the cost or effort for the function is derived. Finally the function estimates are integrated to produce one complete estimate for the whole project.
- The expected-value or three-pint value can be computed for the estimation variable size S, as a weighted average of the optimistic size (S_{opt}), most likely size (S_m), and pessimistic size (S_{pess}) estimates. Consider the following example,

$$S = (S_{opt} + 4S_m + S_{pess}) / 6 \quad ...(5.7.1)$$

- The above example gives the emphasis on "most likely" estimate and it follows a beta probability distribution.

6.15.2.2 An Example of LOC-Based Estimation

- As an example of LOC and FP problem-based estimation techniques, consider a software package to be developed for a computer-aided design application for mechanical components.
- The software is to execute on an engineering workstation and must interface with various peripherals including a mouse, digitizer, high-resolution colour display, and laser printer. A preliminary statement of software scope can be developed.

Table 6.15.1 : Estimation Tables for the LOC Methods

Function	Estimated LOC
User Interface and Control Facilities (UICF)	2,300
Two-Dimensional Geometric Analysis (2DGA)	5,300
Three-Dimensional Geometric Analysis (3DGA)	6,800
Database Management (DBM)	3,350
Computer Graphics Display Facilities (CGDF)	4,950
Peripheral Control Function (PCF)	2,100
Design Analysis Modules (DAM)	8,400
Estimated lines of code	**33,200**

For our purposes, we assume that further refinement has occurred and that the major software functions listed in Table 6.15.1 are identified. Following the decomposition technique for LOC, an estimation table, shown in Table 6.15.1, is developed. A range of LOC estimates is developed for each function. **For example :** The range of LOC estimates for the 3D geometric analysis function is optimistic 4600 LOC, most likely 6900 LOC, and pessimistic 8600 LOC. Applying Equation (1), the expected value for the 3D geometric analysis function is 6800 LOC.

6.15.2.3 An Example of FP-Based Estimation

Table 6.15.2 : Estimating information domain values

Information domain value	Opt.	Likely	Pess.	Est. count	Weight	FP count
Number of external inputs	20	24	30	24	4	97
Number of external outputs	12	15	22	16	5	78
Number of external inquiries	16	22	28	22	5	88
Number of internal logical files	4	4	5	4	10	42
Number of external interface files	2	2	3	2	7	15
Count total						320

Decomposition for FP-based estimation focuses on information domain values rather than software functions. Referring to the table presented in Table 6.15.2, the project planner estimates external inputs, external outputs, external inquiries, internal logical files, and external interface files for the CAD software. FP are computed using the technique discussed For the purposes of this estimate, the complexity weighting factor is assumed to be average. Table 6.15.2 presents the results of this estimate.

Each of the complexity weighting factors is estimated and the value adjustment factor is computed.

	Factor	Value
1.	Backup and recovery	4
2.	Data communications	2
3.	Distributed processing	0
4.	Performance critical	4
5.	Existing operating environment	3
6.	On-line data entry	4
7.	Input transaction over multiple screens	5
8.	ILFs updated online	3
9.	Information domain values complex	5
10.	Internal processing complex	5
11.	Code designed for reuse	4
12.	Conversion/installation in design	3
13.	Multiple installations	5
14.	Application designed for change	5
	Value adjustment factor	**1.17**

Finally, the estimated number of FP is derived :

$$FP_{estimated} = count - total \times [0.65 + 0.01 \sum(F_i)]$$

$$FP_{estimated} = 375$$

The organizational average productivity for systems of this type is 6.5 FP/pm.

6.15.3 Cost Estimation Tools and Techniques

- Estimating size or resources is one of the most important topics in software engineering and IT. You will not deliver according to expectations if you don't plan, and you cannot plan if you don't know the underlying dependencies and estimates.
- An estimate is a quantitative assessment of a future endeavor's likely cost or outcome. People typically use it to forecast a project's cost, size, resources, effort, or duration.
- Today's software market, with its increasing reliance on external components and adapted code, has led to new kinds of technologies for estimation, a practice that has moved from mere size-based approximations to functional and component estimates.
- Standards are starting to evolve as well, because these estimates play such a crucial role in business and enormous amounts of money are at stake.
- Unfortunately, people often confuse estimates with goals or plans. For instance, they schedule projects according to needs instead of feasibility, or commit to something "very urgent and important" before checking how this "urgency" relates to current commitments and capacity planning.
- In fact, most failures in software projects come from belatedly understanding and considering the important difference between goals, estimates, and plans.
- Two of the most important ingredients for proper estimation are people and historical data. They are interrelated more than you might expect because most organizations lack historical data, thus they form estimates primarily through analogy and experience.
- It works if you have experienced people who periodically measure and put their estimates versus actual data into a historical database. Tools can help reducing the time and costs involved in data gathering, as well as supporting reports, risk management, and scenario analysis.
- Following are some Software Estimation Tools :
  - **SLIM-Estimate :** It uses a proven top-down approach that minimizes the input information required to produce fact based, defensible estimates. In addition to software cost estimation, SLIM-Estimate's high level of configurability accommodates the many different design processes used by developers.
  - **SystemStar :** It offers two types of models:
- COCOMO - Software Cost Estimation Model
- COSYSMO - Systems Engineering Cost Estimation Model

6.15.4 Typical Problems with IT Cost Estimates

- Every project proposal needs realistic cost estimates. Similarly, every project manager is dependent on realistic cost estimates to allow for successful cost management.
- Budgeting and cost control is very critical but also challenging under uncertainty. Uncertainty means we do not have all information about the future, and assumptions we make today may come out differently in reality as the project progress.
- Complexity is a fundamental issue here. Simple systems are easy to understand and represent in a model. Thus, simple projects are no challenge in an analytical sense, although they too can be prone to misunderstandings and mistakes.

- The complex system is a different matter altogether. There are four types of complexity :
  - **Structural complexity :** Seeing how projects fit together and how interdependencies pose risk and uncertainties.
  - **Technical complexity :** Maturity of technology and how problems are solved through the design of processes or products.
  - **Directional complexity :** Alignment of people's objectives and motivation, and
  - **Temporal complexity :** Bringing on project parts or components at the right time and the handling of changes, especially in design, as well as cultural understanding of time.
- Project management (PM) is traditionally considered a means of planning and control of activities producing a unique product or service. A major challenge in planning is realistic cost estimation.
- This is true in IT-development projects based on a comprehensive review of previous cost estimation papers and articles.

❑❑❑

UNIT IV

CHAPTER 7

Agile Development Process

Syllabus

Agile Development : Agile manifesto, agility and cost of change, agility principles, myth of planned development, toolset for the agile process.

Syllabus Topic : Agile Development Process

7.1 Agile Development Process

- In today's modern world, businesses spread across the globe in each sphere of life. It has become the global phenomenon. Due to this changing environment, the client must respond to new opportunities and to the volatile market conditions, and to the arrival of new products and services.
- The software application is actually the heart of all the business operations. So according to the rapid changing rules and business ideas, it should respond quickly. Thus the developer is supposed to develop the new software quickly to meet the desired requirements in the stipulated time.
- In fact, in such rapidly changing market condition, rapid development of the software and its urgent delivery is the need of the time and it is supposed to be the most critical requirement of any software system.
- Normally most of the clients are even ready to compromise the quality of software and the requirements at the cost of faster development of the software and its quick delivery.
- Since all the businesses usually operate in the rapidly changing environment, it becomes highly impossible to gather complete set of requirements. The requirements elicited by the customers in the beginning are always changed since the customers find it difficult to predict the actual working of the software and the challenges to come across.
- Thus understanding the requirements quickly and accommodate the requirements change narrated by the customer quickly is the need of the time. Otherwise, delayed delivery of the software may make it unusable and out of date.
- Thus waiting for the complete requirements gathering from the customer will not work for the rapid development. Since the requirement of the customer changes with the progress of the software development process.
- Due to this reason, the conventional approaches like waterfall model or specification based processes will delay the final delivery of the software system.
- But in some cases like critical control system, the complete analysis of the

requirement is mandatory else it may cost life. For such type of safety systems, plan-driven approach is always the best choice.

- Rapid software development processes are the ultimate choice for the development of useful products in quick time span. In this approach the software is developed as a series of increments.
- Agile methods are incremental development methods in which the increments are usually small the new versions of the system are created rapidly and handed over to the customers within the span of 15 to 20 days and their feedback is received for the next versions.

7.1.1 Agile Methods

- In the early days of software development, developer used to plan the project carefully, focus on quality assurance and employ the analysis and design methods supported by various CASE tools.
- This was the general practice of the software developer community that was responsible for the development of larger projects such as government projects etc.
- This larger projects use larger team and the team members may spread geographically across the world and they worked on the software for the longer period of time.
- Therefore, the development period may extend up to the 10 years from the initial specification to the final delivery of the product.
- Such a planed-driven approach will involve significant overheads in all the stages of the development processes like requirement specification, planning, designing, coding and documentation.
- In order to overcome this heavy weight plan-driven approach, the concept of agile methods was proposed. This methodology mainly focused on the product instead of focusing on the design and documentation part.
- All the agile methods trust on the incremental approach instead of the conventional waterfall approach. The incremental approach is the best approach where the customer requirements and the software requirements change rapidly or frequently.
- The philosophy behind agile methods is also observed in **Agile Manifesto** that is accepted by most of the leading software developers. The Agile Manifesto is discussed in the following section.

7.1.2 Agile Manifesto SPPU - May 15

University Question

Q. What is Agile manifesto? **(May 2015, 5 Marks)**

- The Agile Manifesto states that :

  We are uncovering better ways of developing software by doing it and helping others do it.

- Through this work we have come to value :
  - **Individuals and interactions** work over processes and tools
  - **Working software** takes responsibility of comprehensive documentation
  - **Customer collaboration** look after contract negotiation
  - **Responding to change** after having a good plan.
- Even though these agile methods are based on the concept of incremental development, quicker delivery and faster deployment, the agile manifesto gives different processes to achieve this.
- But most of the set of principles used by various experts that are based on agile manifesto are very much common.

Syllabus Topic : Agility and Cost of Change

7.2 Agility and Cost of Change

- Agility can be defined as
  - Effective response to change
  - Effective communication among stakeholders
  - Drawing customers onto team
  - Organizing team so that it is in control of the work performed
- In order to deliver agile software it includes following things
  - It depends what is required by customer
  - Recognizes plans are short lived
  - Develops software iteratively with focus on construction activity
  - Delivers multiple software increments.
  - Adapts to changes.

7.2.1 Cost of Change

- Cost of change curve is shown in Fig. 7.2.1.
- This cost curve is for single production release.
- We need to consider cost of change for different production release.

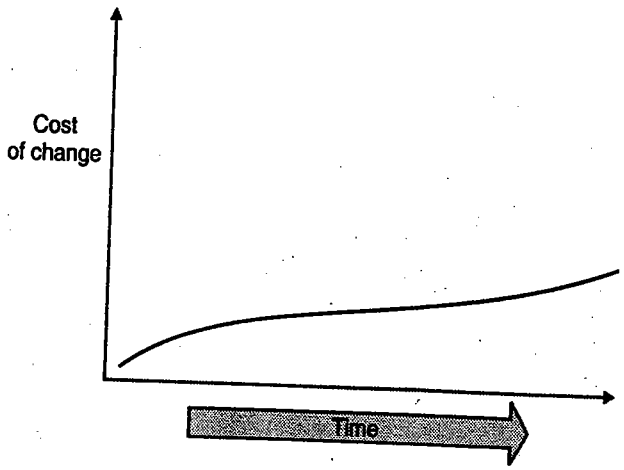


Fig. 7.2.1 : cost change in agile development

Syllabus Topic : Agility Principles

7.3 Agility Principles SPPU - Dec.14

University Question

Q. Discuss agility principles used in agile development. **(Dec. 2014, 5 Marks)**

Review Question

Q. What are the different agility principles ?

- An agile process model includes the concept of development along with a set of guidelines necessary for the development process. The conceptual design is necessary for the satisfaction of the customer. The concept is also necessary for the incremental delivery of the product. The concept or the philosophy makes development task simple. The agility team must be highly motivated, updated with latest skill sets and should focus on development simplicity.
- We can say that agile development is an alternative approach to the conventional development in certain projects.
- The development guidelines emphasize on analysis and design activities and continuous communication between developers and customers. An agile team quickly responds to changes. The changes may be in development, changes in the team members, and changes due to new technology. The agility can be applied to any software process. It emphasizes rapid delivery of operational software.
- The agile alliance has given twelve agility principles as follows :
  1. Satisfy customer through early and continuous delivery.
  2. Accommodate changing requirements.
  3. Deliver the working software frequently in shorter time span.
  4. The customer, business people and developers must work together on daily basis during entire development.

5. Complete the task with motivated developers and facilitate healthy and friendly environment.
6. Convey the message orally, face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile process promotes sustainable development.
9. Achieve technical excellence and good design.
10. There must be simplicity in development.
11. The best architecture, requirements and design emerge from self-organizing teams.
12. Every team should think how to become more effective. It should review regularly and adjust its behaviour accordingly.

- Thus the principles of agility may be applied to any software process. To achieve agile development, the team must obey all the principles mentioned above and conduct proper planning.
- All the agile software processes should address three important assumptions :
  - Difficult to predict in advance software requirements
  - Design and construction are interleaved in most of the projects, it is difficult to predict design before construction.
  - Analysis, design, construction and testing are not much predictable.
- To address these assumptions of unpredictability, the agile development process must be adaptable. So an agile process must adapt incrementally.

Syllabus Topic : Myth of Planned Development

7.4 Myth of Planned Development

There are different myths in the agile development. Different myths are discussed below.

- **Agile development is methodology :** It is not methodology. It is set of values and principles which guide as a set of development methods. Different methods includes Adaptive software development, Agile modeling, Agile unified process, Crystal method, Disciplined agile delivery, Dynamic system development method, Extreme programming, Feature driven development, Learn software development, etc.
- **Agile is undisciplined :** Actually agile focus on individuals and interaction compared to process and tools. This is reason some user misinterprets agile process is undisciplined.
- **Agile has no planning :** Planning is everywhere. Planning can inhibit agility.
- **Agile has no documentation :** Main objective of agility is to remove documentation. It does not mean totally removing the documentation. Agility focus on working software than the documentation. This is because of the dynamic nature of software development.
- **Agile has no upfront design :** It traces on simplified design. Developer generally implements only needed features.
- **Agile does not scale :** scaling software development is difficult. Some believe agile software will work for small projects but not for large projects. Agile process usually

break large, complex projects into small modules. It means it is scalable.

- **Agile is just another fad :** agile process is used over the years which mean we cannot say it is fad. Fad means for time being and short-lived.

Syllabus Topic : Toolset for the Agile Process

7.5 Toolset for the Agile Process

Following are most commonly used toolset for agile processes :

(1) JIRA

(2) Kanban

7.5.1 JIRA

- JIRA is an agile tool used for issue tracking and project management by over 25,000 customers in 122 countries in the world.
- JIRA lets you prioritize, assign, track, report and audit your issues from software bugs and help-desk tickets to project tasks and change requests.
- JIRA is offered in three packages :
  - o JIRA Core includes the base software.
  - o JIRA Software is intended for use by software development teams and includes JIRA Core and JIRA Agile.
  - o JIRA Service Desk is intended for use by IT or business service desks.
- The main features of JIRA for agile software development are the functionality to plan development iterations, the iteration reports and the bug tracking functionality.
- JIRA is a commercial software product that can be licensed for running on-premises or available as a hosted application. Pricing depends on the maximum number of users.

7.5.1.1 JIRA platform

- Every JIRA application (JIRA Software, JIRA Service Desk, and JIRA Core) is built on the JIRA platform. The JIRA platform provides a set of base functionality that is shared across all JIRA applications, like issues, workflows, search, email, and more.
- You can extend and modify this functionality via the integration points provided by the JIRA platform, including the JIRA REST APIs, webhooks, plugin modules, etc.
- The pages in this section will help you learn about the JIRA platform and how to develop for it. You'll find information on JIRA's architecture and JIRA add-ons, guides to developing for different parts of JIRA, JIRA Data Centre, and more.

7.5.1.2 JIRA Architecture

- JIRA is a web application written in Java. It is deployed as a standard Java WAR file into a java Servlet Container such as Tomcat.
- JIRA uses Open Symphony's WebWork 1 to process web requests submitted by users. Please note that WebWork 1, not 2, is used. WebWork 1 is a MVC framework similar to Struts. Each request is handled by a WebWork action which usually uses other objects, such as utility and Manager classes to accomplish a task.
- JIRA uses JSP for the View layer. So most of HTML that is served to the user as the response to their web request is generated by a JSP. Therefore, to generate a response the WebWork action uses a JSP.

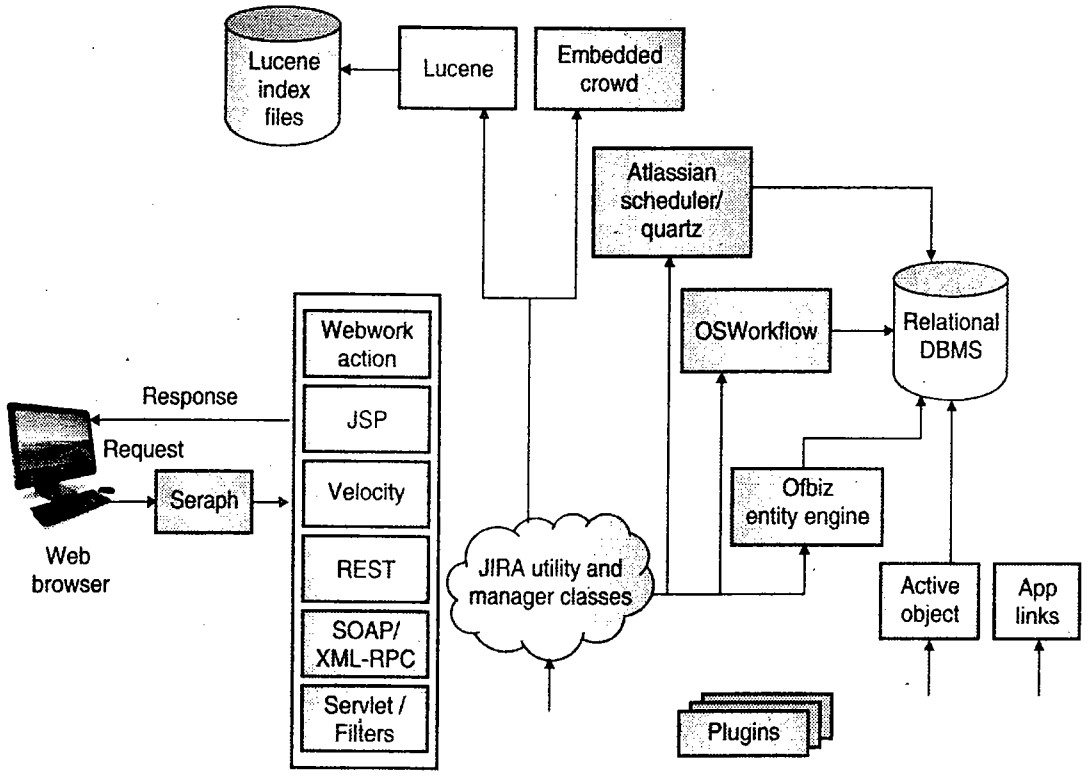


Fig. 7.5.1 : JIRA architecture

7.5.1.3 JIRA Database Schema

- To generate schema information for the JIRA database, e.g. PDF file, follow the instructions below. You can generate schema information in pdf, txt and dot formats. Note, if you want to generate the schema in PDF format, you need to have Graphviz installed.

  (1) Download the attached plugin:
  - For JIRA 5 : JIRA-schema-diagram-generator-plugin-1.0.jar
  - For JIRA 6 : JIRA-schema-diagram-generator-plugin-1.0.1.jar
  - For JIRA 7 : JIRA-schema-diagram-generator-plugin-1.1.0.jar

  (2) Install the plugin in your JIRA instance by following the instructions on Managing JIRA's Plugins.
  - Go to the JIRA administration console and navigate to System > Troubleshooting and Support > Generate Schema Diagram
  - (tick)Keyboard shortcut: g + g + start typing generate
  - Enter the tables/columns to omit from the generated schema information, if desired.
  - If you want to generate a pdf, enter the path to the Graphviz executable.
  - Click Generate Schema.
  - The 'Database Schema' page will be displayed with links to the schema file in txt, dot and pdf format.

7.5.1.4 JIRA Mobile Connect

JIRA Mobile Connect (JMC) is an iOS library that can be embedded into any iOS app to provide following extra functionality:

1. **Real-time crash reporting**, have users or testers submit crash reports directly to your JIRA instance.
2. **User or tester feedback** views that allow users or testers to create bug reports within your app.

3. **Rich data input,** users can attach and annotate screenshots, leave a voice message, and have their location sent.
4. **Two-way communication with users,** thank your users or testers for providing feedback on your app.

7.5.1.5 JIRA Applications

- The JIRA family of applications currently consists of JIRA Software, JIRA Service Desk, and JIRA Core. A JIRA application is built on the JIRA platform, and extends and modifies the base functionality provided by the JIRA platform.
- For example, the JIRA Software application provides software development features that are not part of the JIRA platform, such as agile boards, linked development tool information (e.g. commits, builds, etc), and software project templates.

7.5.1.6 JIRA APIs

- The JIRA platform provides both Java APIs and REST APIs that you can use to interact with JIRA programmatically. These APIs are common to all JIRA applications.
- In addition, JIRA Software and JIRA Service Desk provide APIs for application-specific functionality. For example, the JIRA Software REST API has methods for creating sprints, creating boards, retrieving epics, etc.

7.5.2 Kanban

- Kanban is a popular framework used by software teams practicing agile software development. It is enormously prominent among today's agile software teams, but the kanban methodology of work dates back more than 50 years.
- In the late 1940s Toyota began optimizing its engineering processes based on the same model that supermarkets were using to stock their shelves.
- Supermarkets stock just enough products to meet consumer demand, a practice that optimizes the flow between the supermarket and the consumer.
- Because inventory levels match consumption patterns, the supermarket gains significant efficiency in inventory management by decreasing the amount of excess stock it must hold at any given team.
- Meanwhile, the supermarket can still ensure that the given product a consumer needs is always in stock.
- Agile software development teams today are able to leverage these same JIT principles by matching the amount of work in progress (WIP) to the team's capacity.
- This gives teams more flexible planning options, faster output, clearer focus, and transparency throughout the development cycle.
- While the core principles of the framework are timeless and applicable to almost any industry, software development teams have found particular success with the agile practice.
- In part, this is because software teams can begin practicing with little to no overhead once they understand the basic principles.
- Unlike implementing kanban on a factory floor, which would involve changes to physical processes and the addition of substantial materials, the only physical things a software teams need are a board and cards, and even those can be virtual.

7.5.2.1 Kanban Boards

- The work of all kanban teams revolves around a kanban board, a tool used to visualize work and optimize the flow of the work among the team.

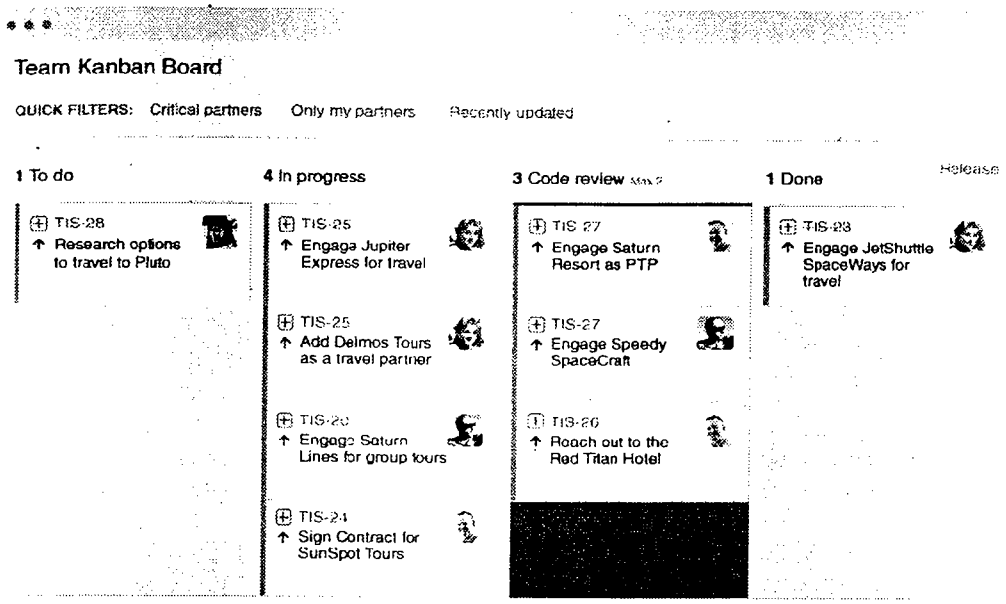


Fig. 7.5.2 : Kanban boards

- While physical boards are popular among some teams, virtual boards are a crucial feature in any agile software development tool for their traceability, easier collaboration, and accessibility from multiple locations.
- Regardless of whether a team's board is physical or digital, their function is to ensure the team's work is visualized, their workflow is standardized, and all blockers and dependencies are immediately identified and resolved.
- A basic kanban board has a three-step workflow: To Do, In Progress, and Done. However, depending on a team's size, structure, and objectives, the workflow can be mapped to meet the unique process of any particular team.
- The kanban methodology relies upon full transparency of work and real-time communication of capacity, therefore the kanban board should be seen as the single source of truth for the team's work.

7.5.2.2 Kanban Cards

- In Japanese, kanban literally translates to "visual signal." For kanban teams, every work item is represented as a separate card on the board.
- The main purpose of representing work as a card on the kanban board is to allow team members to track the progress of work through its workflow in a highly visual manner.
- Kanban cards feature critical information about that particular work item, giving the entire team full visibility into who is responsible for that item of work, a brief description of the job being done, how long that piece of work is estimated to take, and so on.
- Cards on virtual kanban boards will often also feature screenshots and other technical details that is valuable to the assignee.
- Allowing team members to see the state of every work item at any given point in time, as well as all of the associated details, ensures increased focus, full traceability, and fast identification of blockers and dependencies.

7.5.2.3 The Benefits of Kanban

- Kanban is one of the most popular software development methodologies adopted by agile teams today. Kanban offers several additional advantages to task planning and throughput for teams of all sizes.

- Planning flexibility : A kanban team is only focused on the work that's actively in progress. Once the team completes a work item, they pluck the next work item off the top of the backlog.
- The product owner is free to reprioritize work in the backlog without disrupting the team, because any changes outside the current work items don't impact the team.
- As long as the product owner keeps the most important work items on top of the backlog, the development team is assured they are delivering maximum value back to the business. So there's no need for the fixed-length iterations you find in scrum.
- Shortened cycle times : Cycle time is a key metric for kanban teams. Cycle time is the amount of time it takes for a unit of work to travel through the team's workflow–from the moment work starts to the moment it ships.
- By optimizing cycle time, the team can confidently forecast the delivery of future work.
- Overlapping skill sets lead to smaller cycle times. When only one person holds a skill set, that person becomes a bottleneck in the workflow. So teams employ basic best practices like code review and mentoring help to spread knowledge.
- Shared skills mean that team members can take on heterogeneous work, which further optimizes cycle time. It also means that if there is a backup of work, the entire team can swarm on it to get the process flowing smoothly again.
- For instance, testing isn't only done by QA engineers. Developers pitch in, too.
- In a kanban framework, it's the entire team's responsibility to ensure work is moving smoothly through the process.
- Fewer bottlenecks : Multitasking kills efficiency. The more work items in flight at any given time, the more context switching, which hinders their path to completion.
- That's why a key tenant of kanban is to limit the amount of work in progress (WIP). Work-in-progress limits highlight bottlenecks and backups in the team's process due to lack of focus, people, or skill sets.
- For example, a typical software team might have four workflow states: To Do, In Progress, Code Review, and Done. They could choose to set a WIP limit of 2 for the code review state.
- That might seem like a low limit, but there's good reason for it: developers often prefer to write new code, rather than spend time reviewing someone else's work.
- A low limit encourages the team to pay special attention to issues in the review state, and to review others work before raising their own code reviews. This ultimately reduces the overall cycle time.
- Visual metrics : One of the core values is a strong focus on continually improving team efficiency and effectiveness with every iteration of work.
- Charts provide a visual mechanism for teams to ensure they're continuing to improve. When the team can see data, it's easier to spot bottlenecks in the process (and remove them).
- Two common reports kanban teams use are control charts and cumulative flow diagrams.
- A control chart shows the cycle time for each issue as well as a rolling average for the team.

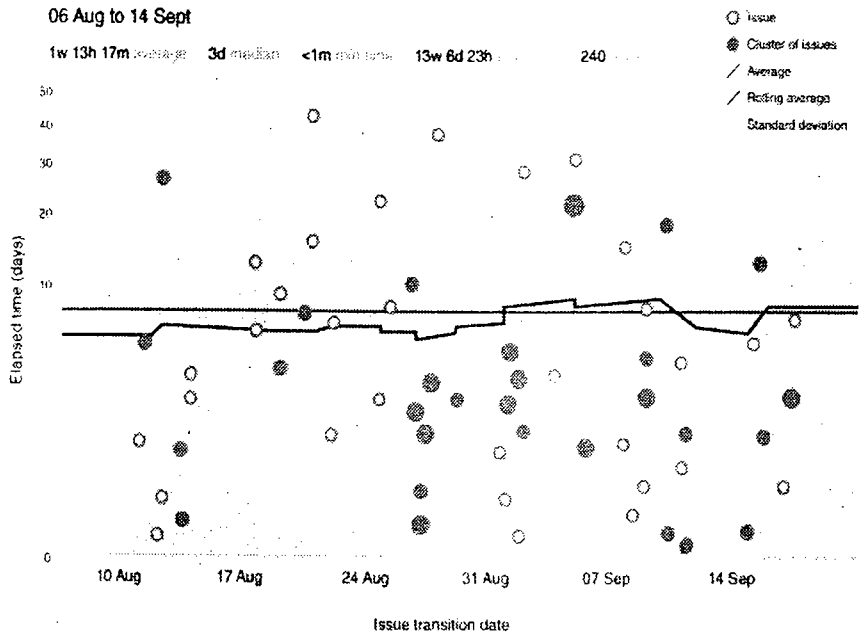


Fig. 7.5.3 : Control chart

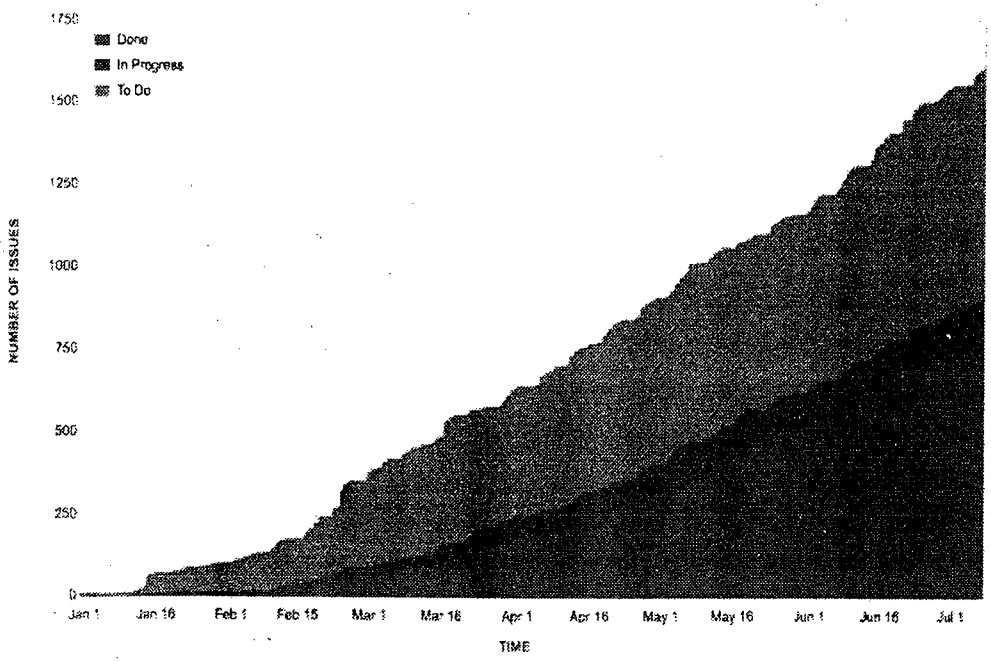


Fig. 7.5.4 : Cumulative flow diagram

- A cumulative flow diagram shows the number of issues in each state. The team can easily spot blockages by seeing the number of issues increase in any given state. Issues in intermediate states such as "In Progress" or "In Review" are not yet shipped to customers, and a blockage in these states can increase the likelihood of massive integration conflicts when the work does get merged upstream.

- Continuous delivery : We know that continuous integration–the practice of automatically building and testing code incrementally throughout the day–is essential for maintaining quality. Now it's time to meet continuous delivery (CD). CD is the practice of releasing work to customers frequently–even daily or hourly. Kanban and CD beautifully complement each other because both techniques focus

- on the just-in-time (and one-at-a-time) delivery of value.
- The faster a team can deliver innovation to market, the more competitive their product will be in the marketplace. And kanban teams focus on exactly that: optimizing the flow of work out to customers.

7.5.2.4 Comparison between Kanban and Scrum

Kanban	Scrum
In Kanban, there is continuous flow.	In Scrum approach, we have regular fixed length sprints (i.e. 2 weeks)
Continuous delivery or at the team's discretion.	Delivery at the end of each sprint if approved by the product owner.
Cycle time is the key metrics in Kanban approach.	Velocity is the key metrics.
Changes can happen at any time.	Teams should strive to not make changes to the sprint forecast during the sprint. Doing so compromises learning around estimation.
Roles : No existing roles. Some teams enlist the help of an agile coach.	Roles : Product owner, scrum master, development team etc.

❑❑❑

UNIT IV

CHAPTER 8

Extreme Programming

Syllabus

Extreme Programming: XP values, process, industrial XP, SCRUM - process flow, scrum roles, scrum cycle description, product backlog, sprint planning meeting, sprint backlog, sprint execution, daily scrum meeting, maintaining sprint backlog and burn-down chart, sprint review and retrospective.

Agile Practices: test driven development, refactoring, pair programming, continuous integration, exploratory testing versus scripted testing

Syllabus Topic : Extreme Programming

8.1 Extreme Programming (XP)

SPPU - May 13, May 15, Dec. 15

University Questions

Q. Explain in detail extreme programming. **(May 2013, 5 Marks)**

Q. What is extreme programming ? **(May 2015, 5 Marks)**

The Extreme Programming is one of the most commonly used agile process models. All the agile process models obey the principles of agility and the manifesto of agile software development.

- The XP uses the concept of object oriented programming. This approach is preferred development paradigm.
- As in conventional approach, a developer focuses on the framework activities like planning, design, coding and testing, the XP also has set of rules and practices.
- Following Fig. 8.1.1 shows the Extreme Programming process and all the key XP activities are summarized.

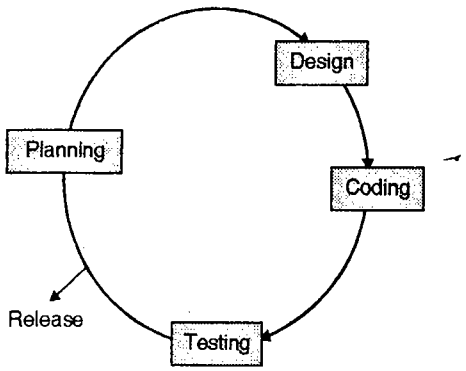


Fig. 8.1.1 : Extreme programming process

8.1.1 XP Values

SPPU - May 15, Dec. 16

University Questions

Q. List the drivers which are treated as XP values. **(May 2015, 5 Marks)**

Q. Discuss XP values. **(Dec. 2016, 5 Marks)**

- Following are a set of five values that establish a foundation for all work performed in context with XP.
  1. Communication
  2. Simplicity
  3. Feedback
  4. Courage
  5. Respect

- For any development process, there must be a regular meeting of developer and the customer. There should be a proper communication for requirement gathering and discussion of the concepts.
- The simple design can always be easily implemented in code.
- The feedback is an important activity which leads to the discipline in the development process.
- In every development projects, there is always a pressure situation. The courage or the discipline will definitely make the task easy.
- In addition to all the XP values, the agile should inculcate respect among all the team members, between other stake holder and with customers.

8.1.2 The XP Process SPPU - Dec. 16

University Question

Q. Discuss XP process. (Dec. 2016, 5 Marks)

- The XP process encompasses four framework activities :

(i) Planning	(ii) Design
(iii) Coding	(iv) Testing

- **Planning** starts with requirement gathering activity that enables XP team to understand the business rules for the software. Customers and developers work together to decide the final requirement.
- **Design** of the product follows the KIS (Keep It Simple) in the XP environment. A simple design is always welcome over the complicated ones.
- **Coding** starts after the design work is over. Once the code is completed, it can be unit-tested immediately. The important concept during the coding activity in XP recommends that two people work together at one computer to create the code.
- **Testing** - All the individual unit tests are organized into a 'universal testing suite'. Integration and validation testing of the system can occur on daily basis. XP acceptances test, also called customer tests are specified by customer and focus on overall system features and functionality.

8.1.3 Industrial XP (IXP)

- Industrial extreme programming is defined as : "IXP is organic evolution of XP. It is customer-centric and filled with test – driven sprit." It differ most from the original XP in its greater inclusion of management, its expanded role for customers and its upgraded technical practices".
- IXP included six new practices that are designed to help XP projects.
  (i) Readiness assessment
  (ii) Project community
  (iii) Project chartering
  (iv) Test-driven management
  (v) Retrospectives
  (vi) Continuous learning
- Readiness assessment takes care that whether appropriate development environment exists to support IXP or not.
- The agile team should have the right candidates to ensure success.
- The IXP team itself will assess the project to determine the appropriate business justification.
- Test-driven management establishes a series of tests to find out the bugs and flaws.
- An IXP team conducts a specialized technical review after a software increment is delivered called retrospective.

- Continuous learning is essential as it is a vital part of continuous process improvement.

8.1.4 The XP Debate

- The critics have raised many issues with reference to XP practices and its evolution.
- Some of the issues discussed or debated are :
  - Requirements volatility.
  - Conflicting customer needs.
  - Requirements are expressed informally.
  - Lack of formal design.
- Due to the active involvement of customer in the agile team, changes are requested in an informal manner. This leads to most of the problems in the development phase. This is called as requirement volatility.
- Since in many projects, there are multiple customers with their own set of needs. This leads to conflicts in customer's need.
- Since requirements are expressed informally, it is difficult to build a system which should not have the inconsistencies.
- The extreme programming always prefers that design of all kinds should be informal. Due to this the design of complex systems can not ensure quality and maintainability. So XP lacks formal design, which is very important.

Syllabus Topic : SCRUM

8.2 SCRUM

- It is one of the agile software development methods. It is an iterative and incremental software development framework. It gives holistic and flexible development strategies.
- It enables teams to self organize by online collaborations of team members. Its key principle is to recognize that during project development customers can change their requirement and requirements of customers are unpredictable.
- It adopts empirical approach. It assumes problem cannot be fully understood or defined but focusing on maximizing team ability to deliver quickly.
- Different terminologies used in SCRUM process is as follows :
  - **SCRUM team :** It contains product owner, development team and scrum master.
  - **Product owner :** Person responsible for product backlog.
  - **SCRUM master :** Person responsible for scrum process.
  - **Development team :** Team of people responsible for delivering the product.
  - Sprint burn down chart : Daily sprint progress.
  - **Release burn down chart :** Chart of completed product backlog.
  - **Product backlog :** List of priority wise requirement.
  - **Sprint backlog :** List of task to be completed which are prioritized.
  - **Sprint :** Period in which development occurs.
  - **Spike** : Period used to research concept.
  - **Tracer bullet :** It is spike with current architecture, technology, best practices, etc.
  - **Tasks :** Items added to sprint backlog at the beginning of sprint which are broken into hours.
  - **Definition of done :** Exit criteria which decides product backlog items are completed or not.
  - **Velocity :** Total efforts a team is capable of in a sprint.
  - **Scrum-but :** It is exception to scrum methodology.

8.2.1 Process Flow

SCRUM process flow contains different stages. Different stages are as follows :

➢ **Setup**

Setup environment contains the following :

- Remove any customization from existing application.
- Activate scrum process pack plug-in.
- Assign scrum roles to users.

➢ **Create a product**

- Product represents the functionality which is identified by owner.
- Product contains different features which are needed for enhancement which are useful for customer satisfaction.
- It can have few features or many features.

➢ **Create user stories**

- These are product requirement created by product owner.
- User stories are written in plain language. Less technical jargons are used.
- It contains specific requirement that product should have.
- Stories cannot be created without associating with product.

➢ **Create release**

It has start and end date in which number of development iterations are completed.

➢ **Create sprint**

- It is basic unit of time in development process.
- It can have any length. Typically it takes one to four weeks to complete.
- Scrum master creates one or more sprints.
- Sprints should release in given start and end dates.
- Scrum team need to complete all the stories which are committed.
- Scrum master expects all the stories are fully implemented and ready to release.

➢ **Plan sprint**

- Team and scrum master need to decide on which stories they can commit to complete within a sprint.
- Scrum master make sure that capability of sprint team matches the requirement of stories.
- If capacity of the stories exceeds then scrum master add team members, remove stories or add sprint as and when needed.
- Velocity chart is used in estimation process.
- Velocity chart shows the historical record of number of completed points.
- With the help of velocity chart scrum master get the idea of capacity of team.
- Velocity chart is important tool in planning sprint.

➢ **Track sprint progress**

- Scrum master is responsible for checking the progress.
- He provides the progress report of the team.
- Team members frequently update task and records.

➢ **Generate charts**

- Progress of a sprint can be tracked with the help of different charts.

- Chart is one of the important tools of scrum team.
- Burn down chart compares ideal progress in sprint against the actual progress.
- It is done on daily basis.

8.2.2 Scrum Roles

Different roles are used in scrum process.

Three different roles are as follows :

1. Product owner
2. Development team
3. Scrum master

1. Product owner

- They are stakeholders of the customers.
- They ensure teams delivers values to business.
- They write customer centric items, rank customer centric item and add to product backlog.
- The scrum process needs to be one product owner. Sometimes they are part of development team.
- Role of the product owners in product requirement are as follows :
  - Important role of product owner is communication with development team.
  - Identifying important requirement and communicating is important task of product owner.
  - They generally reduce the communication gap between team and stakeholders.
  - They demonstrate the solution to stakeholders.
  - They announce the different release of the product.
  - They communicate the team status.
  - Organizes the milestone review.
  - They educate the stakeholders in development process.
  - They negotiate priorities, scope, funding and schedule.

2. Development team

- They are responsible for delivering the product.
- Team generally consists of people of different skills.
- Team size can be from between 3 to 9.
- They includes member like analyst, designer, developer, tester, technical communicator, document writing, etc.
- Development team is self organizing.

3. Scrum master

- Scrum is facilitated by scrum master.
- He is accountable for product goals and deliverables.
- He is not a team lead or manager. He acts as a buffer between development team and distracting influences.
- He is the enforcer of the scrum process.
- He generally involves in key meeting and challenges the team to improve.
- Project manager is responsible for people management but scrum master is not related to people management.

8.2.3 Scrum Cycle Description

- Scrum cycle is divided into different activities.

- Scrum cycle activities are as follows :

1. Define
2. Plan
3. Build
4. Review
5. Retrospect

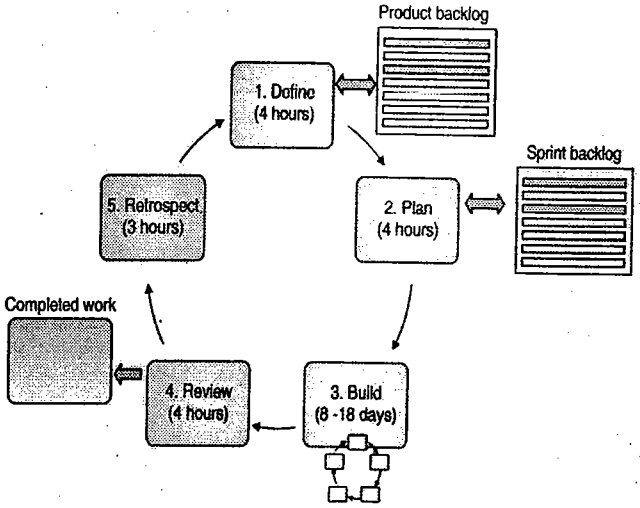


Fig. 8.2.1 : SCRUM cycle

1. **Define** : During defining the requirement generally scrum team, scrum master and product owner sits together. They decide the priority of team for duration of sprint. It is useful in high level direction for the team.
2. **Plan :** It is about selecting items from product backlog. It also evaluates value of different items and estimate the efforts needed by scrum team. Generally items are selected from product backlog which can be considered as sprint backlog.
3. **Build :** Task from the sprint backlog is selected for development. They go on selecting the items till the items from the sprint backlog get completed.
4. **Review** : Review is presented by scrum team to product owner about the implementation of the items.
5. **Retrospect :** It is final steps of the retrospection. It has few objectives. It focuses on identifying the improvement area by the scrum team. They also discuss on the success of earlier iterations.

8.2.4 Product Backlog

- It is ordered list of requirements which are maintained for product.
- It contains different details like features, bug fixes and non functional requirements.
- It contains whatever is needed for the success of the product.
- Items from the product backlog are ordered by product owner using different factors which includes date needed to complete the task, business values, risk, dependencies, etc.
- Product backlogs are written in story format.
- It mainly contains what exactly need to be delivered.
- It also contains the order in which sequence the requirement should be delivered.
- Requirement listing and ordering that requirement is solely done by product owner.
- It contains assessment of business values by the product owner.
- Development team's assessment is also part of product backlog.
- Different methods are used by product owner to decide the priority of requirement which includes Fibonacci sequence and other methods.
- If some requirements are urgent then product owner can request to scrum team about prior requirement.
- Sizes of backlog items are determined by the development team.
- It does not contain only user stories, but it contains other information also.
- Product owner is responsible for maximizing the value of the product. He is also responsible for maximizing the work of development team.

- Product backlog is used for :
  - Taking request for product modification.
  - It includes adding new facility, replace old facility, remove some unwanted features.
  - Ensure delivery team is given work which maximizes the business benefits to the owner of the product.

8.2.5 Sprint Planning Meeting

- Product owner, development team and scrum master is involved in the sprint planning meeting.
- If required outside beneficiary can be invited for sprint planning meeting.
- In this meeting product owner describes the highest priority work and communicate to development team.
- Development team ask the question if any doubts, to the product owner.
- This meet is held at the beginning of sprint cycle.
- They generally select what work need to be done.
- They prepare sprint backlog which gives details of time taken to complete the work.
- They discuss how much work should be done during the current sprint cycle.
- Maximum time limit is 8 hours.
- Within 8 hours, in the first four hour entire team discusses about prioritizing the product backlog.
- In second 4 hours, development team plan for sprint which comes with sprint backlog.
- Sample agenda for sprint planning meeting is as follows :
  - Product roadmap
  - Status of development
  - Name of the iterations
  - Velocity of previous iterations
  - Capacity of team
  - Concerns
  - Review
  - Updates
  - Stories
  - Tasking out
  - Commit
- Sprint planning meeting template is shown in Fig. 8.2.2.

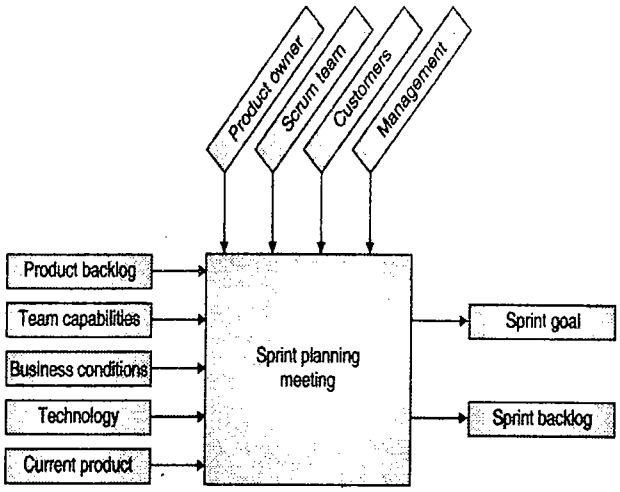


Fig. 8.2.2 : Sprint planning meeting

8.2.6 Sprint Backlog

- It is nothing but list of work development team need to complete in the sprint.
- This list is derived from product backlog.
- It contains list of user stories in sprint in the order of priority.
- It also contains the relative effort estimate.
- The task needed to develop every story is also maintained in the sprint backlog.
- In sprint planning meeting team selects some number of product backlog items.
- Product backlog items are selected in user story format.
- They also identifies user stories need to complete.
- Size of sprint backlog is selected by team.
- Sprint backlog can be maintained using spreadsheet.

- Example of sprint backlog is shown below.

Table 8.2.1

User Story	Tasks	Day 1	Day 2	Day 3	Day 4	Day 5	....
As a member, I can read profiles of other members so that I can find someone to date.	Code the ...	8	4	8	0		
	Design the ...	16	12	10	4		
	Meet with Mary about...	8	16	16	11		
	Design the UI	12	6	0	0		
	Automate tests...	4	4	1	0		
	Code the other ...	8	8	8	8		
As a member, I can update my billing information	Update security tests	6	6	4	0		
	Design a solution to ...	12	6	0	0		
	Write test plan	8	8	4	0		
	Automate tests...	12	12	10	6		
	Code the ...	8	8	8	4		

8.2.7 Sprint Execution

- Sprint is basic unit of development in scrum development.
- Scrum make progress is series of sprint.
- During the sprint every day project status meeting occurs. This meeting is called as daily scrum meeting.
- Sprint execution is nothing but the work which is performed by scrum team to meet sprint goals.
- All the work which is necessary to deliver is performed.
- It accounts for majority of time during a sprint.
- It begins after sprint planning.
- It ends when sprint review starts.

8.2.8 Daily Scrum Meeting

- Every day during sprint, project meeting occurs.
- This meeting is called as daily scrum meeting.
- It is helpful for schedule coming day's work.
- Scrum meeting has following guidelines :
  - All members presents for meeting with updates.
  - Usually meeting starts on time even if some members are absent.
  - Usual meeting time is at the morning.
  - Meeting time and venue are same every day.
  - Approximate meeting time is 15 minutes.
  - Generally core team member speak in the meeting.
- Different questions are answered by team. Some of the questions are as follows :
  - What have been done since yesterday?
  - What is today's planning?
  - Any impediments or stumbling blocks?
- Sample scrum meeting is shown below :

Table 8.2.2

Monday	Tuesday	Wednesday	Thursday	Friday
		Sprint 1 planning 1 pm. - 5 pm.	Daily scrum 9:15 - 9:30	Day scrum 9:15 - 0:30
Daily scrum 9:15 - 9:30	Daily scrum 9:15 - 9:30 Backlog grooming 1 pm - 2 pm	Daily scrum 9:15 - 9:30	Daily scrum 9:15 - 9:30 Backlog grooming 1 pm - 2 pm	Daily scrum 9:15 - 9:30
Daily scrum 9:15 - 9:30	Daily scrum 9:15 - 9:30 Dry run and prep for sprint review (optional) 2 pm - 3 pm	Daily scrum 9:15 - 9:30 Sprint 1 review 10 am - 11 am Sprint 1 retrospective 11 am - 12 pm Celebration lunch 12 pm - 1 pm Sprint 2 planning 1 pm - 5 pm		

8.2.9 Maintaining Sprint Backlog and Burn-Down Chart

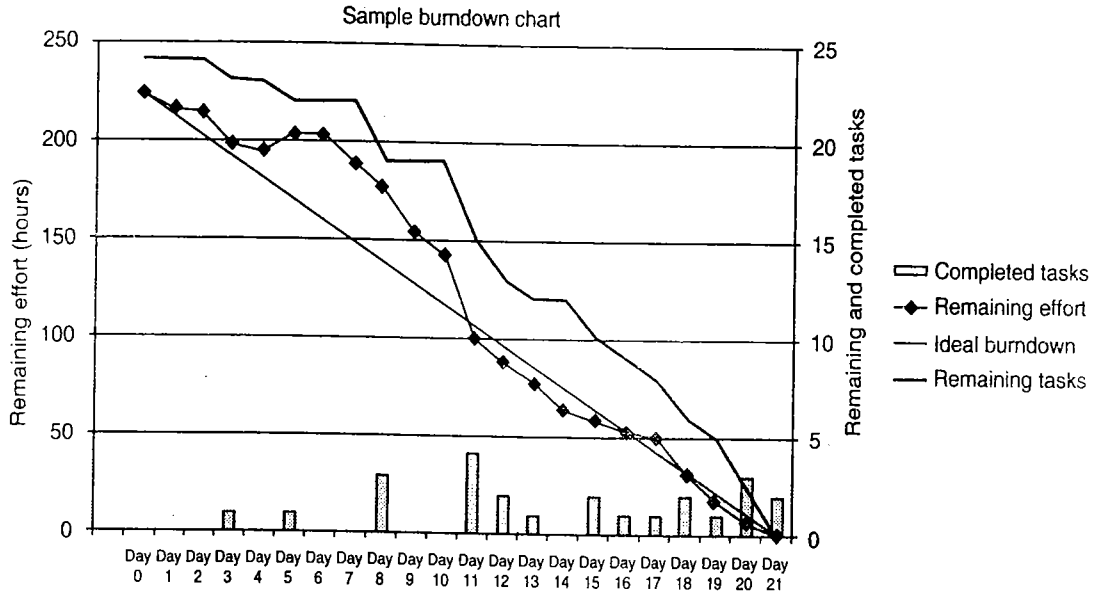


Fig. 8.2.3 : burn down chart

- Burn down is graphical representation of work left versus time.
- Pending work is on vertical axis and time is at horizontal axis.
- It is nothing but the run chart of outstanding work.
- It is useful in predicting when the work will be completed.
- It is used in agile software development particularly in scrum process.
- It can be applicable to any project.
- Sample burn down chart is shown in Fig. 8.2.3.

8.2.10 Sprint Review and Retrospective

- These meetings are held at the end of sprint cycle.
- These meetings are sprint review and retrospective meeting.
- In the sprint review meeting following points are discussed :
  - Review the completed work.
  - Review incomplete work.
  - Show the completed work to stakeholders.
  - Incomplete work not demonstrated.
  - Meeting is limited to four hours.
- In the sprint retrospective following points are considered :
  - Every team member reflects on past sprint.
  - Make continuous process improvement.
  - Two questions are asked in the sprint. One question is : What is well in the current sprint? What improvements can be done in next sprint?
  - The time limit for the sprint retrospective meeting is 3 hours.
  - This meeting is head by scrum master.

Syllabus Topic : Agile Practices

8.3 Agile Practices

Agile development is supported by number of practices. It includes the different area like requirement analysis, design, coding, testing, project management, process, quality, etc. Some of the important agile practices are as follows :

- Pair Programming
- Refactoring
- Test Driven Development
- Continuous Integration

8.3.1 Pair Programming SPPU - Dec.15

University Question

Q. Describe XP concept of pair programming. **(Dec.2015, 5 Marks)**

Review Questions

Q. Explain Pair programming in the context of agile software development.

Q. What are various benefits of pair programming ?

Q. What do you mean by Remote pair programming ?

- It is one of agile practice is agile software development.
- In this method two programmer work together on one machine.
- One programmer types the code. Another programmer observes point and suggests the changes.
- Sometimes programmer can switch their role to each other.
- Observer programmer can suggest improvement in code.
- With this method programmer who writes the code can concentrate on efficient coding. Observer programming can think and suggest improvements.

Benefits of pair programming

There are numerous advantages of pair programming.

- **Economy :** It spends around 15% more time on programming. But designed code is efficient than that of individual coding. It also improves development time and support cost.
- **Design quality :** With pair design more ideas can be generated. As two programmers are working together we are able to check all possibilities during design. This can be accomplished as different programmer bring their prior experience together. They think differently and able to generate new ideas.
- **Satisfaction :** With pair programming programmer can enjoy work than that of individual programming. They are more confident in their solutions. Their confidence results in successful software.
- **Learning :** Ideas are shared among the programmer. They are able to learn from each other. They are able to share their knowledge with each other. It allows programmer to observe programming code and give the feedback accordingly.
- **Team building and communication :** It allows to share the problem with each other and able to find the solution on it. With this practice communication among team member increase with benefits in greater team building.

Remote pair programming

- It is part of pair programming.
- It allows programmer to be geographically apart from each other but connected through the network.
- It is also called as virtual pair programming or distributed pair programming.
- They generally work using shared editors, shared desktop, remote pair programming tools.
- Sometimes this method creates the problems which are not present in face to face programming.
- Some of the problem is remote pair programming are delay in communication, communication link failure problem, lack of coordination, etc.

- Different tools are provided for remote pair programming. Some of the tools are as follows :
  - Microsoft Lync
  - VNC
  - Screenhero
  - Skype
  - Terminal multiplexers (tmux a, screen -x)
  - Gobby
  - SarosXpairtise
  - Floobits
  - Visual studio anywhere
  - Cloud

8.3.2 Refactoring SPPU- Dec. 15, Dec. 16

University Questions

Q. Describe XP concepts of refactoring. **(Dec. 2015, 5 Marks)**

Q. Write short note on refactoring. **(Dec. 2016, 5 Marks)**

Review Questions

Q. Explain Refactoring in the context of agile software development. What are different refactoring techniques ?

Q. What are different editors and IDE that supports automated refactoring ?

- It is one of the agile practices in agile software development.
- It is process of restructuring the existing code. The changes are made without affecting external behaviour.
- It does not change the functional requirement of the code.
- It changes the non-functional requirement of the code. Non functional requirements are useful in efficiency, performance, throughput, etc.
- It has numerous advantages like improved readability, reducing complexity, reusability, etc.
- It results in creating more expressive software architecture which results in extensibility.
- Generally series of standards are applied for code refactoring.
- Some standards are used which are designed by organization and some other standards also used in code refactoring.
- They are generally identified by code smell.
- E.g. we have code which is very long or it is almost duplicate of another code. In this situation code refactoring can be applied where we can remove the code duplication. After refactoring the basic functionality of the code remain same.
- Code duplication can be removed by designing shared function. Shared by will be used whenever required instead of creating same copy of code again and again.
- Two main objectives of refactoring are maintainability and extensibility.
- With code maintainability, we can easily identify the bugs in the software as the code is properly documented.
- With code extensibility, new features can be added very easily.
- We need automatic unit test before applying code refactoring.

List of refactoring techniques

- Some of famous refactoring techniques we will discuss in this topic.
- Some techniques are applied to only certain languages or situations.
- Many development environments provide code refactoring.
- Some of the techniques are as follows.
- **Techniques allowing more abstraction**
  - Code encapsulation where different fields can be accessed with getter and setter methods.
  - Code sharing can be achieved by generalizing types. If we make generalize code then chances of sharing that code increases.

- Replace type checking code with states
- Use polymorphism.

- **Techniques for breaking code apart into more logical pieces**
  - Break the code into different modules. Modules increase the code reusability.
  - Extracting class results in moving code to new code.
  - Extracting methods results into moving methods to new methods.
- **Techniques for improving name and location of code**
  - Move methods or field to appropriate class.
  - Change the name of methods or fields to user friendly name. Name should signify the purpose of method or variable.
  - Use concept of superclass from object oriented programming.
  - Use concept of subclass from object oriented programming.

Automated code refactoring

Some editors and IDE supports automated refactoring. Some of the software are listed below.

– IntelliJ IDEA	– WebStorm	– Eclipse
– Netbeans	– JDeveloper	– Visual studio
– ReSharper	– Code rush	– Visual assist
– Photran	– Xcode	– AppCode

8.3.3 Test Driven Development (TDD)

Review Questions

Q. Write short note on test driven development.

Q. Explain test driven development (TDD) with block diagram. What are different phases of TDD ?

- It is one of the agile practices in agile development.
- This software development process relies on repetition of short development cycles.
- Changes in software are quick and easy.
- Short development cycle includes writing automated test case which defines desired improvement or new function. Minimum coding is done to pass the test. Afterwards code are redesigned as per the standard.
- It is related to extreme programming.
- Test driven development is shown in Fig. 8.3.1.

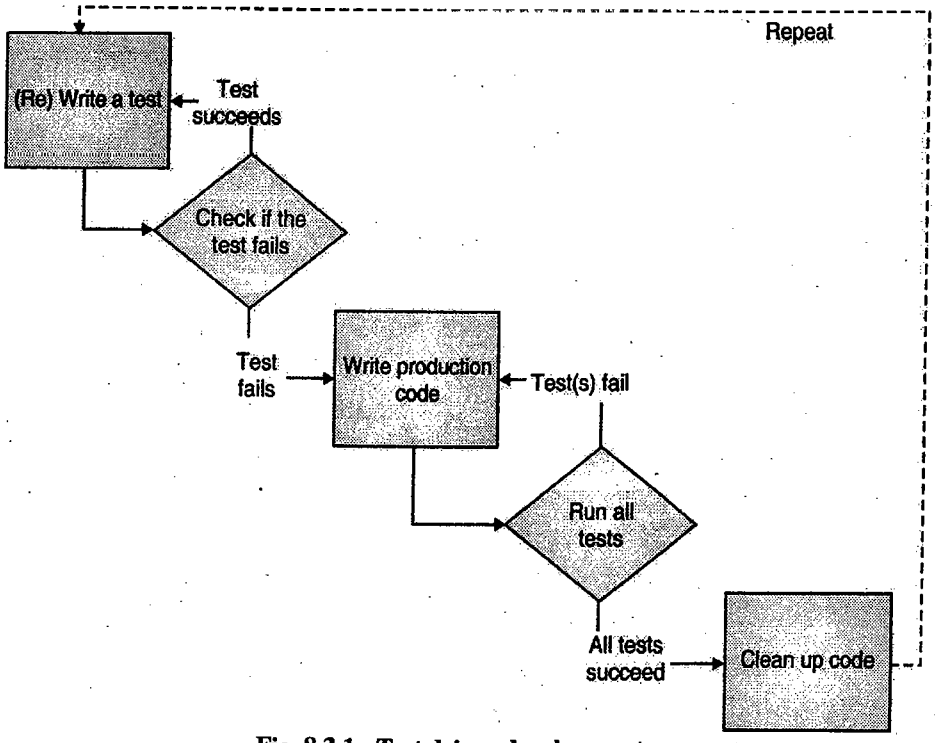


Fig. 8.3.1 : Test driven development

- Different phases of test driven development are as follows.

(a) Create test

- In test driven development new feature can be added by creating test.
- This test should get fail because this feature was not added previously.
- Before writing test, before should understand the test properly then only test should be added.
- Requirements can be understood using different techniques like use cases and user stories.

(b) Run test and see it fails

- In this phase we run the created test and check if it running properly or not.
- Test should get fail so that we can write code for it.

(c) Writing some code

- In this phase we write code for test to get pass.
- The code written is not perfect but it is useful to pass the test for time being.
- Afterwards this code will be improved.
- Purpose of this code is just for passing the test and not for permanent purpose.

(d) Run test

We run the test on created code.

(e) Refactor code

- Earlier we have written code which was not final.
- In this phase, code will improved and made efficient to fulfill all the coding standard.

(f) Repeat

- The process is repeated for another test also.
- Size of steps should be small.
- If the test fails then programmer need to undo the changes.

8.3.4 Continuous Integration

Review Questions

Q. Explain Continuous integration in the context of agile software development. What are various principles of Continuous integration?

Q. List different advantages of Continuous integration.

Q. List Software tools which supports continuous integration.

- It is method of merging all developer copies of work together frequently.
- It is part of extreme programming.
- Main aim is to prevent integration problems.
- It is useful in iterative software development.
- Earlier it was used in coordination with test driven development.

Principles of continuous integration

- **Maintain code repository :** All the project source code should be properly stored in code repository. It is useful in revision control system.
- **Automate the build :** There should be facility for building the integration automatically. Single command should able to do all the activities. Different build tools like make can be used. Automated build includes integration.
- **Make build self testing :** After making the integration software should work as per specification. Testing facility after integration should be automated one.

- **Everyone commits to baseline every day :** With this facility every committer can reduce conflicting changes.
- **Every commit should be built :** system should build commits to current working version.
- **Keep the build fast :** Building process should be rapid.
- Other principles are as follows.
  - o Test in clone of the production environment
  - o Make it easy to get the latest deliverables
  - o Everyone can see the results of the latest build
  - o Automate deployment

Advantages

It has many advantages which are as follows.

- If unit test fails, developer can revert the code to bug free state of the code.
- Developers can detect and fix integration problem continuously.
- Early warning of broken or incomplete code.
- They can give early warning of conflicting changes.
- Code is available constantly.
- Immediate feedback can be given to developers
- Creating modules can be possible with continuous integration.

Software tools which supports continuous integration

- Apache Continuum
- Apache Gump
- AutomatedQA Automated Build Studio
- Atlassian Software Systems Bamboo
- Buildbot
- BuildHive
- CABIE
- Cascade
- CruiseControl
- CruiseControl.rb
- CruiseControl.NET
- Electric Cloud
- FinalBuilder Server
- Hudson
- Jenkins
- IBM Rational Team Concert
- IBM Rational Software SCLM
- TeamCity
- TinderboxTravis CI
- Xcode 5

8.3.5 Exploratory Testing Versus Scripted Testing

SPPU - Dec. 14, May 16, Dec. 16

University Questions

Q. Compare scripted testing verses exploratory testing. **(Dec. 2014, 5 Marks)**

Q. Differentiate between exploratory testing and scripted testing. **(May 2016, Dec. 2016, 5 Marks)**

Review Question

Q. Compare Exploratory testing with scripted testing in the context of agile software development.

- Scripted testing considers two roles namely test designer and tester. Test designer is usually designs the test cases. He is high skilled specialist. Tester executes the test case which is designed by test designer.
- In exploratory testing different roles are not considered. Test designing and actual testing is done by same person. Tester designs the test cases and executes it. According to previous results he may create more test case and execute it.
- Learning, test design and execution all are done by tester in exploratory testing.
- In script testing team can be formed of different employees where one of the members would be very high skilled one and other members can beginners. Cost of testing can be saved with script testing.
- In exploratory testing depends upon skill of the employees which results in high project cost.

- With scripted testing high planning is possible. We can predict for desired outcome with scripted testing. As scenarios are ready we execute the test cases precisely.
- In exploratory testing planning is impossible or very difficult. We cannot guarantee all the test cases will be executed or not.
- With scripted testing well documentation can be created. Test designer can document the test scenarios whereas tester can records the status of executed test. With exploratory very less documentation is used.
- Exploratory testing is flexible whereas script testing is not.
- If some changes happen in the requirement then test designer need to change the scenarios. When scenarios are changed then only tester can executes the test cases.
- Exploratory testing is interesting than that of scripted testing.

8.4 Agile Modeling (AM)

Review Question

Q. Discuss the significance of Agile Modeling (AM). What are different modeling principles that make agile modeling unique ?

- There are many situations where the software developer build very large software systems used for some business applications and other critical software system. The software developer should model the complexity of such system so that he should be able to define the scope of the system. The software developer do so for the following reasons :
  - All the component of the system should be understood properly.
  - The actual problem can be divided into various partition to solve and merge it into a single unit.
  - The quality of the software is evaluated at each step during its construction.
- In the last three decades, various modelling technique have been put forwarded for analysis and design at both architectural level and component level.
- All these methods have profound impact and are difficult to apply. This method once applied is difficult to sustain over the life of software project.
- In order to handle this deficiency, some rigid techniques are required so that the projects can be handled intellectually.
- The best alternative to this approach is agile modelling. The agile modelling is defined as the practice based methodology that is very effective in modelling documentation of various software systems.
- The agile modelling is actually a collection of design principles and practices for modelling and it is applied to software system.
- In actual practice, the agile models are more effective and lightweight as compared to traditional models.
- In addition to agile manifesto the agile modelling suggest following techniques :
  - **Model with a purpose :** A software developer using agile modelling must have a proper and specific goal in his mind before he start developing a project .
  - **Use multiple model :** The software developer is free to use to various model and notation to explain a software in a better way. The agile modelling suggests different aspect of system and provides the value to the intended end-users.
  - **Travel light :** Agile modelling also suggest to use the models that provide

long terms values and must be able to accommodate to the changes occurring at the later stages i.e. the developer must used light weight model.

- **Content is more important than presentation :** In agile modelling the contents are very important for the intended users. Based on presentation only, the work can not be accomplished
- **Know the model and the tools you are using to create the software :** While using agile methodology a software developer must know the strength and weaknesses of each of the models and tools.
- **Adapt locally :** The agile Modelling approach must be adaptable to the needs of the team.

❑❑❑

UNIT V

CHAPTER 9

Project Monitoring and Control

Syllabus

Project monitoring and control: tools for project management, Software tools like Microsoft project management or any other open source tools.

Syllabus Topic : The Management Spectrum

9.1 The Management Spectrum

SPPU - May 12, Dec. 12, May 15

University Questions

Q. What are the 4 Ps involved in software project management? **(May 2015, May 2014, 5 Marks)**

Q. What is the relevance of four Ps in project planning ? Explain in detail.**(Dec. 2012, 12 Marks)**

- Management spectrum for effective software focuses on the four P's :

1. People
2. Product
3. Process
4. Project

- How these four P's can be used in the management spectrum? It is interesting to note that the order can never be arbitrary.

9.1.1 The People

SPPU - Dec. 12

University Question

Q. Explain the term people of management Spectrum. **(Dec. 2012, 3 Marks)**

- Software engineering institute has developed model named the People Management Capability Maturity Model (PM-CMM). The PM-CMM has been developed to deploy the talent needed to improve their software development capability.
- Not only to deploy talent but also to enhance the readiness of software organizations to undertake increasingly complex applications by helping to grow, attract, motivate the talent needed to improve their software development capability and to continue to hold the talent to improve the development capabilities.
- Also helping to retain the talents that are needed to improve the development capabilities.
- The PM-CMM defines the following key practice areas for software people :
  - Recruiting
  - Selection
  - Performance management
  - Training
  - Compensation
  - Career development

- Organization and work design
- Team or culture development

- The PM-CMM is very close to the software capability maturity model integration. It guides organization in creation of a mature software process.
- There are various groups that are involved for the most needed communication and co-ordination issues required for the effective software. All these groups can be categorized as under :

1. Stake holders
2. Team leaders
3. Software team
4. Agile teams
5. Co-ordination and communication issues

9.1.1.1 Stake Holders

Stake holders are the people who are directly or indirectly involved in the software process and software project. To make the effective development process, the project team must be organized in such a way that maximizes each person's skills and capabilities. This is the job of the team leader that looks into each and every corner of the process. The stake holders can be categorized into one of five categories required :

1. Senior managers
2. Project managers
3. Developers
4. Customers
5. End users

1. **Senior managers** have lot of influence on the project and they are the people who define business issues.
2. **Project managers** can organize, plan motivate and control the developers who develop the software project. Actually, all these are the technical people.
3. **Developers** have the technical skills that are important to engineer a product or application.
4. **Customers** are one of the important stakeholders who specify the requirements. These requirements are later developed. Also some other stakeholders are also there who have interest in the outcome of the software.
5. Finally the **end-users** who are going to use the software.

9.1.1.2 Team Leaders

- Competent practitioners often make poor team leaders because project management is a people-intensive activity. But unfortunately in many cases individuals just fall into a project manager role and become the project managers accidentally.
- The **MOI** (Model of Leadership) is used by the project managers that has the following characteristics to make a project manager very effective:
  - **Motivation :** The ability to encourage the team to produce their best ability.
  - **Organization :** The ability to organize the ongoing processes in such a way that will help the initial concept to mould into a final product.
  - **Ideas or innovation :** The ability to encourage the team to create the innovative ideas let them feel to be more creative in their tasks.
- All the successful project leaders apply a problem solving management style i.e. a software project manager should be able to concentrate on understanding the problem and finding the solution, managing the flow of ideas and thoughts. At the same time, manger should let everyone in the team to

know that quality is important parameter and can not be compromised.

- One other view of the characteristics that defines an effective project manager should emphasizes following four key factors :
  - **Problem solving :** A project manager should find out technical and organizational issues.
  - **Managerial identity :** A project manager should take the charge of the project. He should have overall control and should allow only good technical team members to do their own styles.
  - **Achievement :** To enhance the productivity of a project team, he should take initiative and demonstrate the function through his own actions.
  - **Influence and team building :** A project manager should have the ability to read the faces and capability of his team members i.e. face reading.

9.1.1.3 Software Team

- The people those are directly involved in a software project are within the project manager's scope. The structure of a good team depends on the management methodology adopted in the organization, the number of people who will inhabit the team and their skill levels, and the overall complexity of the question.
- Following are seven project factors that must be considered when planning the structure of software engineering teams :
  - The ability to solve the difficulty of the problem.
  - The overall size of the final programs in lines of code.
  - The amount of time that the team will stay together.
  - The degree of modularity.
  - The quality and reliability of the system.
  - The delivery date rigidity.
  - The degree of communication required for the project.
- To achieve a high-performance team :
  - Team members must have trust in one another.
  - The distribution of skills must be appropriate to the problem.
  - An unorthodox or independent-minded person may have to be excluded from the team, if team unity is to be maintained.
- The aim and idea for every project manager is to create a team that exhibits unity among the team members.

9.1.1.4 Agile Teams

- The term agile means very active. As a cure to many of the problems that have afflicted software project work, in recent years, agile software development has been put forward.
- The agile philosophy provides the following :
  - It encourages customer satisfaction by early incremental delivery of software.
  - It provides small highly motivated project teams.
  - It consists of informal methods.
  - It also provides minimal software engineering work products.
  - It results in overall development simplicity.
- The small sized and highly motivated project team, also called an agile team, adopts many of the characteristics of successful software project teams.

9.1.1.5 Co-ordination and Communication Issues

There are some factors because of which, software projects get into problems. These factors are based on the following features of modern software :

1. Scale

Since the scale of most of the development efforts is large, therefore it leads to confusion, complexity and significant difficulties in coordinating team members.

2. Uncertainty

Uncertainty is very common and it results in a continuing stream of changes that will increase or decrease the size of project team.

3. Interoperability

- Interoperability is one of the important characteristics of many software applications. The new software should be able to communicate with the existing system and satisfy the predefined constraints and conditions implemented by the software.
- The following important characteristics of software are the facts of life :
  1. Scale
  2. Uncertainty
  3. Interoperability
- To handle these characteristics very effectively, the development team must devise some method for communicating and coordinating the team members.
- The formal communication is achieved through formal meetings like writing and non-interactive communication channels.
- The informal communication is more personal focusing individuals. All the members of a software team must share their ideas on ad hoc basis and must ask for any help required for the problems encountered on a daily basis.

9.1.2 The Product

- The product objectives and scope must be established before a planning a project. All the technical and management constraints and difficulties should be identified and their alternative solutions should be considered.
- Without this information, it is impossible to define reasonable and accurate estimation of the cost, and conduct an effective assessment of risk and a realistic breakdown of project tasks.
- In order to define the product objectives and its scope, there should proper coordination between the software developer and customer and they must meet.
- In most of the cases, this particular activity begins as part of the system engineering or business process engineering and continues as the first step in software requirements engineering.
- Objectives identify the overall goals for the product. These objectives do not take into consideration, how these goals will be achieved. The scope identifies the primary data and functions and the behaviours that characterize the product, and more importantly, attempts to bound these characteristics in a quantitative manner.

9.1.3 The Process SPPU - Dec. 12

University Question

Q. Explain the term process of management Spectrum. (Dec. 2012, 3 Marks)

- A comprehensive plan can be drafted using software process framework in software development process.

- All these framework activities can be applied to all projects either small sized or large sized and regardless of the complexity of software projects
- Following factors enable the framework activities to be adapted to the characteristics of the software project :
  - o The tasks set
  - o Work products
  - o Milestones
  - o Quality assurance points
- And finally, the umbrella activities mentioned below overlay the process model :
  - o SQA (Software Quality Assurance)
  - o SCM (Software Configuration Management)
  - o Measurement
- Generally all these umbrella activities are independent of any one framework activity and they are applied throughout the development process.

9.1.4 The Project

- Planned and controlled software projects are conducted for one and only reason because it is the only way known to manage complexity.
- Project failure rate remains higher than it should be even though the success rate for software projects has improved. One of the reasons is that the customers lose interest quickly (because what they have requested was not really as important as they first thought), and the projects are cancelled.
- The software engineers and the software project manager must follow certain guidelines in order to avoid project failure :
  - o Heed a set of common warning signs
  - o Understand the critical success factors that lead to good project management and
  - o Develop a common sense approach for planning, monitoring, and controlling the project.

9.2 Project Monitoring and Control

- Monitoring and Controlling a project is the process or activities whereby the project manager tracks, reviews and revises the project activities in order to ensure the project creates the deliverables in accordance with the project objectives.
- Because of the unique and temporary nature of projects, they require active control. Unlike a process where the same set of activities have been performed repeatedly so that habits and expectations are stable, a project is inherently unstable.
- The activities are unique to the project or the sequence of activities and resources are only temporarily assigned and associated with the project and are redeployed when the project completes. Habits and patterns are not established before everything changes.
- The primary results of the Monitoring and Controlling processes are the project performance reports and implementing project changes. The focus for project management is the analysis of project performance to determine whether a change is needed in the plan for the remaining project activities to achieve the project goals.
- Almost every project will require a change to the plan at some point in time. Traditional projects are the most stable projects because the requirements and the activities are clear and well understood. Adaptive and Extreme projects are the least stable.
- They require very close control and will

require numerous changes - if for no other reason the project manager will need to refine the activities of later phases based upon the results of early activities.

- Tools and techniques that are used by project managers to conduct the Monitoring and Controlling of a project fall into one of four general categories. The first is the collection of project performance information. Techniques supporting this category are Pulse Meetings, Variance Reports, and Program Reviews.
- The second category is the analysis of the project performance to determine whether a project change is needed. Techniques that are used in this category are Technical Reviews, Project Forecasting and Problem Solving. The third category is reporting on project performance.
- Techniques that support this activity include the use of a Project Management Information System, Management Reviews, and Dashboards. The final category is the management of project change. The technique in this category is the maintenance of a Change Management Log.
- There are two areas of project management tools and techniques that closely support the Monitoring and controlling process but are also used more broadly throughout the project lifecycle.

9.3 Tools for Project Management

- The project planning and project management is an important activity performed by the project managers in order to estimate the following entities for their optimum use :
  - The resources required for the project under development.
  - The cost of the project, and
  - The schedule for in-time completion of the project
- Project Managers can use a range of tools and techniques to develop, monitor and control project schedules.
- These tracking tools can automatically produce critical path diagrams or Gantt charts. These tools also instantly update time plans as soon as new information is entered and produce automatic reports to monitor and control the project.
- These tracking tools can assist in planning and controlling the expenditures more effectively and improve communication of financial reporting.
- These tools also provide automatic report template with good GUI to effectively communicate with all the stakeholders of the project.
- These tools also look into work breakdown structure and risk management with greater accuracy and ease of updating and printing the reports.
- Following are some examples of Schedule Tracking Tools :
  - Microsoft Project
  - Daily Activity Reporting and Tracking (DART)

Syllabus Topic : Software Tools Like Microsoft Project Management of any other Source Tools

9.3.1 Microsoft Project

- We can use Microsoft Project 2010 or later versions to plan team projects, schedule tasks, assign resources, and track changes to data that is stored in Team Services or TFS.
- By using Project, you can access many tools and functions through the simplified graphical menus and Office Ribbon. The Team tab menu, as shown in the following figure, displays the same functions that are available from the Team tab in Excel.

Project team tab ribbon

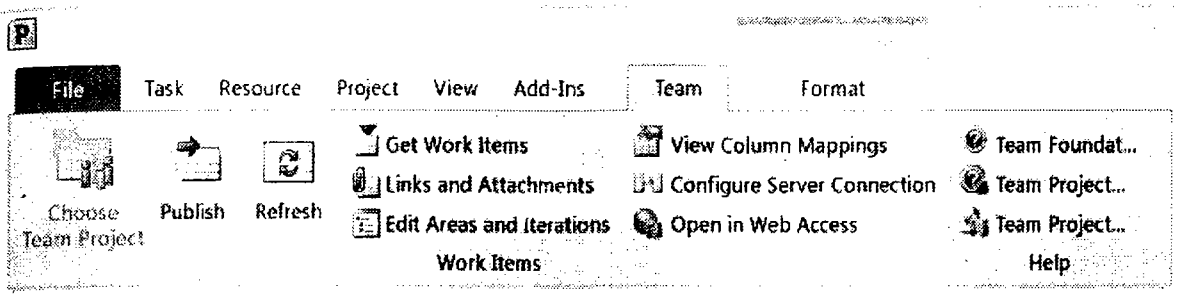


Fig. 9.3.1 : Project team tab ribbon

- Project 2010 and later versions support several new project fields and functions. Depending on how you use Project to schedule team tasks, you may want to update the task work item form to display some of the new fields.
- To maintain new Project fields in both your project plan and in TFS, you have to customize the task work item type and the Microsoft Project Field mapping file for the team project.
- In Project 2010 and Project 2013, New task-related features are felicitated to schedule the tasks manually or automatically.

  By using Task Mode, which is accessed through the following Ribbon menu, you have more flexibility in the way you and team members schedule tasks.

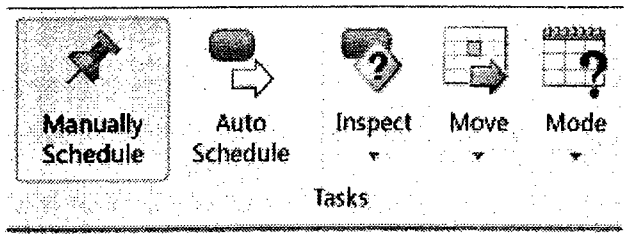


Fig. 9.3.2 : Task Mode

- Start and finish dates for auto scheduled tasks ( ) are determined by the scheduling engine based on task dependencies and the project calendar, as in previous releases of Project.
- Project managers who are accustomed to automatic scheduling with past versions of Project can turn the new manual scheduling feature off for specific tasks or the entire project.
- Finally, it is quite simple to use Microsoft Projects for scheduling the projects. It is quite similar to other products of Microsoft office like Microsoft word, Microsoft excel and others.

9.3.2 Open Source Tool : Daily Activity Reporting and Tracking (DART)

- DART – Daily Activity Reporting Tool, enables you to track the changes to record being made by users. The extension provides the ability to dispatch the email with summary of changes.
- After installation, in the tool we see various tabs. Here we provide brief introduction of these tabs and their usage.

Scheduling

- DART provides cron/modules/DART/DARTCron.service (php file) that can be configured through scheduler for execution at the end-of business day.
- The script will trigger the action to gather the changes and summarize it via email. The email configuration details can be updated in the file modules / DART / DART. Config.php

Example

- Here is a CRON tab entry to schedule the script to be executed every day at 23:00 (i.e., 11:00 PM)

```
0 23 *** php -f /home/site/vtigercrm/cron/modules/DART/DARTCron.service
```

Module Views

When you open DART it shows the list of changes tracked for the day. You can click on Refresh now link if you don't see it updated. The changes are updated when the CRON script is scheduled too.

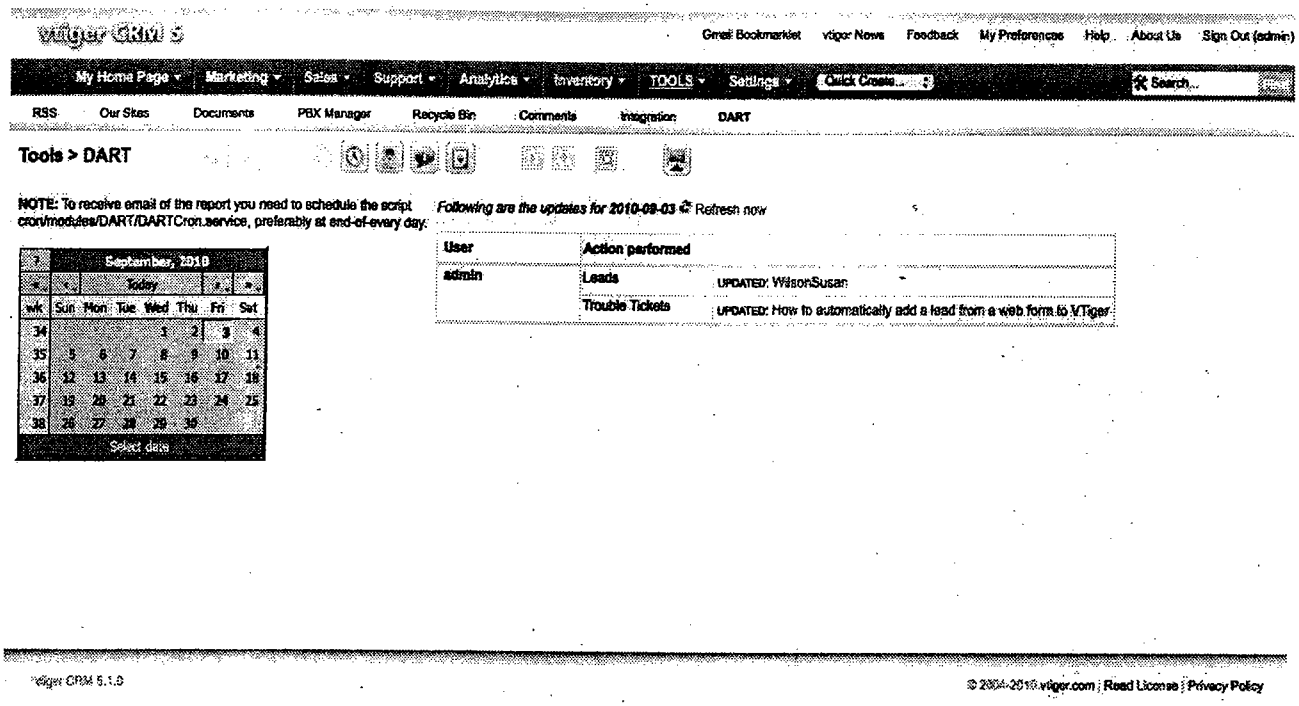


Fig. 9.3.3 : DART Snapshot

No activity notification

- DARTCron.service can send notification emails to group of users who have not used the system for the given day.
- To enable this, you need to uncomment the line in cron/modules/DART/DARTCron.service

```
//$dartCron->sendNoActivityUpdateEmail();
as
$dartCron->sendNoActivityUpdateEmail();
```

❑❑❑

UNIT V

CHAPTER 10

Project Quality Management

Syllabus

The Importance of Project Quality Management: Planning Quality Management, Performing Quality Assurance, Controlling Quality, Tools and Techniques for Quality Control (statistical control, six sigma)

Syllabus Topic : Importance of Project Quality Management

10.1 Importance of Project Quality Management

- Quality control is a process within an organization designed to ensure a set level of quality for the products or services offered by a company.
- Most companies provide a service or a product. The control is important to determine that the output being provided is of overall top quality.
- The Quality Management includes the actions necessary to verify and control the quality output of products and services. The overall goal of quality control includes meeting the customer's requirements, product satisfaction, fiscally sound, and dependable output.
- Most companies provide a service or a product. The control is important to determine that the output being provided is of overall top quality. Quality is important to companies for liability purposes, name recognition or branding, and maintaining a position against the competition in the marketplace.
- Quality control can be implemented with a company in many ways. Some organizations bring in a quality assurance department and practice testing of products before they are delivered to the shelves. When quality assurance is used, a set of requirements is determined and the quality assurance team will verify the product not only meets all of the requirements but they will also perform faulty testing.
- Companies with a customer service department often implement quality controls through recording phone conversations, sending out customer surveys, and requiring employees to follow a specific set of guidelines when speaking to customers over the phone. Implementing a quality control department or strategy allows a company to find faults or problems with products or services before they reach the customer.
- Quality control not only consists of products and services but how well an organization works as a whole together within the organization and in the marketplace.

- A strategy to manage and improve the quality within an organization can help a company become and remain a success.
- Quality is an ongoing effort that must be consistent and improving every day. Every organization or business can benefit by using quality control for their products or services, within the internal organization, and interacting in the marketplace

Syllabus Topic : Planning Quality Management

10.2 Planning Quality Management

- Quality planning is the process of developing a quality plan for a project. The quality plan defines the quality requirements of software and describes how these are to be assessed. The quality plan selects those organizational standards that are appropriate to a particular product and development process.
- An organizational quality plan is typically prepared by the Quality Manager and covers the following issues:
  - **Quality objectives and goals :** The overall objectives of the quality management program together with measurable goals to be achieved
  - **Quality management system scope :** Who and what will be impacted by the quality management system. For example: process scope, product scope, organisational scope
  - **Organisation and responsibilities :** Organisational structure, reporting relationships and roles and responsibilities for quality
  - **Resource requirements :** Identification of the people and resources required to develop, use, maintain and improve the quality management system.
  - **Cost benefit analysis :** A quality program cost benefit analysis addressing issues such as: the cost of poor quality, the cost to improve quality and the cost benefits to be achieved
  - **Activities and deliverables :** The quality management system elements will be produced/improved The quality management system development activities required
  - **Schedule :** The timeframes in which the work will be achieved together with major milestones for quality management system element delivery, review and deployment.
  - **Risk analysis :** An analysis of what could go wrong together with strategies for risk reduction.

Syllabus Topic : Performing Quality Assurance

10.3 Performing Quality Assurance

Review Questions

Q. Explain importance of SQA.

Q. Explain Software Quality Assurance and its importance.

Q. What is Software Quality Assurance ? Explain various factors that affect Software Quality.

Concepts and Definitions

SQA (Software Quality Assurance) is a planned and systematic method for evaluation of the quality of software product, standards, processes, and procedures.

- It assures that the standards and procedures once established are followed throughout the software development life cycle.
- The evaluation of compliance with these standards and procedures is done by continuous monitoring throughout the development process, product evaluation, and by conducting audits.

- Software development and different control processes can also add quality assurance approval points. The evaluation process for SQA can be conducted by using the applicable standards.

Standards and Procedures

- The task of establishing standards and procedures for software development process is quite critical. It is because they provide the framework that is actually a platform for software evolution. The established standards are nothing but the established criteria by which the software engineering products are compared.
- The procedures are also the established criteria by using it, the development processes and different control processes are compared.
- The standards and procedures actually establish the methods for developing software applications. The role of SQA is to provide assurance of their existence and adequacy.
- The documentation of standards and procedures is also very important since all the SQA activities relay on definitions of project compliance. Following are some important SQA activities :
  - o Continuous monitoring
  - o Product evaluation and
  - o Auditing.

10.3.1 Types of Standards

Review Question

Q. Explain Different Quality standards.

1. Documentation Standards
2. Design Standards
3. Code Standards

- The **Documentation Standards** established under SQA define the proper content for planning and control. This documentation standard provides the consistency throughout the development life of a project.
- Similarly the **Design Standards** define the proper form and content of the design product.
- These standards provide rule book for and corresponding methods for translating the software requirements specification into the actual software design and for representing it in the design documentation.
- The **Code Standards** specify the programming language in which the source code has to be written. The code standards also specify various constraints that should be put for usage of the language features. They also specify the language structures, style, patterns, rules for data structures, interfaces, and internal code documentation.
- The procedures must be followed in carrying out a development process. All the development processes must have documented procedures.
- Following are the examples of processes for which procedures are needed :
  - o Configuration management
  - o Non-conformance reports (i.e. any development process not following the SQA standards properly)
  - o Corrective action
  - o Testing and formal inspections.
- If the software products are developed as per the NASA DID, then the management plan defined the software development control processes. For example, configuration management for product standards.

- All the standards should be documented properly as per the **Standards and Guidelines DID** in the product specification.

10.3.2 Software Quality Assurance Activities

Review Questions

Q. List the SQA related activities.

Q. Explain SQA activities.

- The product evaluation and the process monitoring are important SQA activities which give the assurance that the software development processes and the control processes written in the management plan are carried out effectively.
- They also ensure that all the procedures and standards are correctly followed. Products are continuously monitored for checking that it is following the standards and processes. The audits are very important technique to conduct product evaluation and process monitoring.
- The intermittent review of plan also ensures the conformance of standards and procedures laid by SQA activities. The SQA approval points are built directly into these processes.
- Product evaluation is an SQA activity that assures standards are being followed.
- In actual practice, the first products monitored by SQA must be the project's standards and procedures.
- SQA ensures that there are clear and achievable standards and these can evaluate the compliance of the software application product with established standards.
- The product evaluation ensures that the software application product is developed by conforming all the applicable standards as illustrated in the Management Plan.
- The process monitoring activity in an SQA ensures that the appropriate steps are carried out during the development process.
- The SQA standards monitor all the processes by comparing the actual steps carried out with those in the documented procedures.
- The SQA also ensures that the Management Plan specifies the methods that should be used by the monitoring activity of SQA process.
- The basis technique for SQA process is the auditing that looks the entire product and all the processes in depth. This is done by comparing them with the established standards and procedures by the SQA processes.
- The audit is an important activity to review the management plan, technical processes and assurance processes to provide the actual status of the software application product.
- The main idea behind an SQA audit is to ensure that the control procedures are properly followed and the desired documentation is properly maintained. It also ensures that the developer's status reports accurately reflect the status of the activity.
- The SQA product is nothing but an audit report to display the findings and recommendations to force that the development process obeys standards and procedures established.

10.3.3 SQA Relationships to Other Assurance Activities

Following are some of the important relationships of SQA process with management and assurance activities :

1. Configuration Management Monitoring

- SQA ensures that the software Configuration Management (CM) activities are conducted in accordance with the CM plans, standards, and procedures.
- The SQA reviews the Configuration Management plans for compliance with software Configuration Management policies and requirements. It provides the monitoring for non-conformance. The SQA audits the Configuration Management functions for implementing the standards and procedures and prepares the status reports of its findings.
- The Configuration Management activities that are monitored and audited by SQA plan include the following parameters :
  - Baseline control
  - Configuration identification
  - Configuration control
  - Configuration status accounting
  - Configuration authentication.
- The SQA process keeps on monitoring and auditing the software library. The SQA makes sure that the baselines are established and maintained consistently for their use in development and control.
- Software configuration identification (SCI) is consistent and accurate with software programs, modules, units and associated software documents.
- The configuration control is maintained to make it compatible with various critical phases of software development like testing, acceptance and delivery.
- The configuration status accounting is conducted accurately including the recording and reporting of data. It reflects the software's configuration identification and any changes made to the configuration identification, and the implementation status of approved changes.
- The software configuration authentication is established by various configuration reviews and audits. The performance required by the software requirements specification and the configuration of the software is properly maintained in the software design documents.
- The software development libraries give proper code, documentation, and concerned data in their various versions from start of the project to the final delivery of it.
- All the approved changes to baseline software are included in the product consistently and any unauthorized changes are not included.

2. Verification and Validation Monitoring

- The SQA activities ensure Verification and Validation (VandV) activities by using following parameters :
  - Formal technical reviews
  - Time to time inspections and
  - Walkthroughs.
- The SQA roles are observed in the above three parameters i.e. formal technical reviews, inspections, and walkthroughs. The SQA verifies that they are conducted properly.
- The SQA also verifies that any action required is assigned properly and the related documents are maintained and updated properly.
- The FTRs (Formal technical reviews) should be carried out at the end of every life cycle phase. By doing this, the

problems will be detected and corrected. It also checks that all the requirements are satisfied.

- Examples of formal technical reviews are :
  - o PDR (Preliminary Design Review)
  - o CDR (Critical Design Review) and
  - o TRR (Test Readiness Review).
- A technical review takes into consideration overall structure of the software product being developed. All the desired requirements must be fulfilled. The technical reviews are the integral part of the development process.
- In FTR (formal technical reviews), the actual work completed is compared with standards and procedures already established by the software organization. The main objective of SQA is to ensure that all these standards and management plans are executed properly.
- An inspection or the walkthrough are the detailed testing of a product by using step-by-step process and each line of code are traversed to uncover any possible error. The SQA activities ensure that the entire process is properly completed.
- The inspection process is used to follow the compliance to the standards.

3. Formal Test Monitoring

Review Question

Q. Describe Formal Test Monitoring.

- The SQA activities ensure the formal software testing such as acceptance testing. It is carried out in accordance with SQA plans and procedures established earlier.
- The SQA activities review the testing documentation to follow the standards and procedures established.
- The documentation review includes following :
  - o Test plans,
  - o Test specifications,
  - o Test procedures and
  - o Test reports.
- The SQA plan observes the testing and takes continuous follow-up to uncover the non-conformances and correcting them. After the test monitoring, SQA ensures that the final software product is ready for delivery.
- The main objective of SQA monitoring for software testing is to ensure :
  - o The occurrence of any incident during testing are noted and recorded. Usually these incidents are not expected in the test procedures.
  - o All the test reports are accurate and complete.
  - o The regression testing is conducted to evaluate the non-conformances and they are corrected.
  - o The correction of non-conformances takes place happens before the delivery.
  - o The software testing checks that the software satisfies all requirement specifications.
  - o The quality of the testing is ensured. It is done by evaluating that the project requirements are satisfied.
  - o Software Quality Assurance during the Software Acquisition Life Cycle.

- In addition to all the SQA activities mentioned in the earlier sections, there are few phase-specific SQA activities that must be conducted during the Software Life Cycle.
- At the end of each development phase, the SQA concurrence is an important element.

- The suggested activities for each of the development phase are listed below.

1. Software Concept and Initiation Phase

- The SQA activities must be involved in both writing and reviewing the Management Plan. It ensures that the processes, procedures, and standards are appropriate and auditable.
- In this phase, the SQA activities also provide the QA section of the Management Plan.

2. Software Requirements Phase

- In this phase of software development, the SQA plan assures all software requirements are completed and testable.
- It also assures all functional, performance and interface requirements.

3. Software Architectural (Preliminary) Design Phase

The SQA activities in the architectural (preliminary) design phase include following :

- It ensures that the approved design standards are properly followed.
- It ensures that all the software requirements are allocated properly to the software components.
- It ensures that testing verification matrix is available it is updated
- It ensures that the Interface Control Documents are conforming the standards.
- It reviews PDR documentation and ensures that all action items are resolved correctly.
- It ensures that the approved design is placed under configuration management.

3. Software Detailed Design Phase

The SQA activities that are applied during the detailed design phase are as follows :

- It ensures that the approved design standards are followed.
- It ensures that all the allocated modules are present in the detailed design.
- It ensures that the results of design inspections are present in the design.
- It reviews CDR documentation.
- It ensures that all action items are resolved.

5. Software Implementation Phase

The SQA activities that are implemented during the implementation phase consists of following audit :

- The results of coding and design activities are included in the schedule available in the Software Development Plan.
- Checking the status of all deliverable items.

- The configuration management activities.
- The software development library and
- The Non-conformance reports and necessary action taken.

6. Software Integration and Test Phase

The SQA activities in integration and test phase of software development include :

- Ensures readiness for testing of all components.
- Ensures that all tests are run as per test plans and procedures. If any non-conformance is found, then it is resolved.
- Ensures that test reports are completed and in the correct form.
- Once testing is completed, the SQA activities certify it and ensure that it is out for delivery.

7. Software Acceptance and Delivery Phase

The SQA activities applied during the software acceptance and delivery phase are taking care of assurance of the performance of a final configuration audit.

8. Software Sustaining Engineering and Operations Phase

- In this phase, there are mini-development cycles to improve the software product or to correct the software.
- In this development cycle, the SQA activities conduct the appropriate phase-specific activities as mentioned in the above section.

9. Techniques and Tools

- The SQA activities must evaluate its needs for assurance tools in comparison with the available off-the-shelf tools for applicability to the specific project. If it does not match with off-the-shelf (already existing) tools, then it can be developed.
- The audit, automatic code standards analyzers and inspection checklists are some useful tools.

Syllabus Topic : Controlling Quality

10.4 Controlling Quality

- Software Quality Control is the set of procedures used by organizations to ensure that a software product will meet its quality goals at the best value to the customer, and to continually improve the organization's ability to produce software products in the future.
- Software quality control refers to specified functional requirements as well as non-functional requirements such as supportability, performance and usability. It also refers to the ability for software to perform well in unforeseeable scenarios and to keep a relatively low defect rate.
- Following are Quality Control Activities :
  - Check that assumptions and criteria for the selection of data and the different factors related to data are documented.
  - Check for transcription errors in data input and reference.
  - Check the integrity of database files.
  - Check for consistency in data.
  - Check that the movement of inventory data among processing steps is correct.
  - Check for uncertainties in data, database files etc.

- Undertake review of internal documentation.
- Check methodological and data changes resulting in recalculations.
- Undertake completeness checks.
- Compare Results to previous Results.

10.4.1 Comparison between Software Quality Assurance (SQA) and Software Quality Control (SQC)

Criteria	Software Quality Assurance (SQA)	Software Quality Control (SQC)
Definition	SQA is a set of activities for ensuring quality in software engineering processes (that ultimately result in quality in software products). The activities establish and evaluate the processes that produce products.	SQC is a set of activities for ensuring quality in software products. The activities focus on identifying defects in the actual products produced.
Focus	Process focused	Product focused
Orientation	Prevention oriented	Detection oriented
Breadth	Organization wide	Product/project specific
Scope	Relates to all products that will ever be created by a process	Relates to specific product
Activities	Process Definition and Implementation Audits Training	Reviews Testing

Syllabus Topic : Tools and Techniques for Quality Control

10.5 Tools and Techniques for Quality Control

- Quality management and project management are complementary, they both emphasize customer satisfaction. Quality leads to customer satisfaction. The main objective in quality management is making sure that the project meets the needs that it was originally created to meet, nothing more, nothing less. In other words, to ensure quality you must meet the needs of the stakeholder.
- Project quality management consists of three major processes :
  - **Plan quality management**
    Identifying the quality requirements and standards for the project and product (planning process group).
  - **Perform quality assurance**
    Auditing the quality requirements and quality control results to ensure that appropriate quality standards are used (executing process group).
  - **Control quality**
    Monitoring and recording the results of quality activities to assess performance and recommend necessary changes (monitoring and controlling process group).
- The quality management planning process determines the quality standards that are applicable to the project and devising a way to satisfy them. The goal is to create a quality management plan which documents the following :
  - the way the team will implement the quality policy
  - the way the quality of both the project and the product will be assured during the project
  - the resources required to ensure quality
  - the additional activities necessary to carry out the quality plan
- Various tools and techniques are employed on each of the above three major processes :
  - Statistical Control, and
  - Six Sigma

10.5.1 Statistical Control

- Statistical Control sheets are used to organize information in order to facilitate data gathering.

- Check sheets are particularly effective for doing inspections, enabling focus on the particular attributes that may be contributing to potential or identified quality problems.
- Following are some Statistical Control charts :
  - **Pareto Diagrams :** A Pareto diagram is an ordered bar graph showing the number of defects and their causes ranked by frequency.
  - **Histograms :** A histogram is a vertical bar graph that represents the frequency of each measured category (known as bins) of variable.
  - **Control Charts :** Control charts are used to determine if processes are in or out of statistical control. Most processes experience a degree of normal variation (or common cause variation); that is to say, most processes do not achieve target performance all the time.
  - **Scatter Diagrams :** Scatter diagrams plot two variables, the independent variable and the dependent variable, to graphically show the relationship between them.
  - **Benchmarking :** Benchmarking involves comparing the current project or activity to similar projects or activities. This process generates ideas for improvement and provides a standard to measure quality performance.
  - **Statistical Sampling :** Statistical sampling is an approach to selecting a few representative items for inspection and then extrapolating the results to the whole.

10.5.2 Six Sigma

- Six Sigma is a disciplined approach towards process improvement. It was first used by General Electric (GE) Corporation back in 1995.
- Many companies across the globe now are using Six Sigma standards and finding it useful.
- The main purpose behind using Six Sigma is to improve the process for the development of the products faster and reasonable cost.
- The Six Sigma approach uncovers many errors and correct it. Thus we can say that Six Sigma is a systematic approach to achieve perfection. The Six Sigma can be calculated by Six Sigma calculator.
- Basically the Six Sigma is an approach that is based on measurement strategy and obviously focuses on process improvement.
- The Six Sigma has following two sub methodologies :
  - The Six Sigma DMAIC process and
  - The Six Sigma DMADV process.
- The DMAIC stands for define, measure, analyze, improve, control.
- The DMADV stands for define, measure, analyze, design, verify.

□□□

UNIT V

CHAPTER 11

Project Risk Management

Syllabus

The Importance of Project Risk Management, Planning Risk Management, Common Sources of Risk in IT Projects.

Syllabus Topic : The Importance of Project Risk Management

11.1 Risk Management

SPPU - May 12, Dec. 12, May 13

University Questions

Q. Explain risk management process and various types of risks with suitable example. **(May 2012, 6 Marks)**

Q. What are the types of risks? Explain in brief. **(Dec.2012, 6 Marks)**

Q. Explain Software Risk Management. **(May 2013, 6 Marks)**

Review Questions

Q. What are the risks associated with software projects ? How do project managers manage such risks ?

Q. What are the risks associated with project delay?

- Whenever we start any business or any development process, we take into consideration the risks involved in accomplishing that task. Similarly when software development process is started, it is main job of a software manager to look into all types of possible risks involved in the entire process.
- It involves focusing on the possible risks that could affect the project development process :
  - o Schedule of the development process
  - o Quality of the application under construction
- It is the responsibility of a project manager to look into the matter and take necessary action to avoid these risks. All the results of risk analysis and analysis of consequences of risk occurring should be well documented in the planning of the project itself.
- If the risk management is effective, then it becomes easier to handle all the problems and it is ensured that the project schedules and budget are within acceptable limits and there is no schedule slippage and ultimately no budget slippage.
- The always threaten the project development processes and the software under construction.
- Following are three important categories of risk :

1. Project risks
2. Product risks
3. Business risks

1. Project risks

- o These are the risks that directly affect the schedule of the project and the

resources involved in the development process.

- **Example of project loss :** Loss of an experienced developer and designer.

2. Product risks

- The product risks affect the quality and performance of the application built.
- **Example of product risks :** Failure of a purchased component to perform as per expectation.

3. Business risks

- The business risks are affecting the organization those develop and process the software.
- **Example of business risks :** A competitor of the organization introducing a new product.

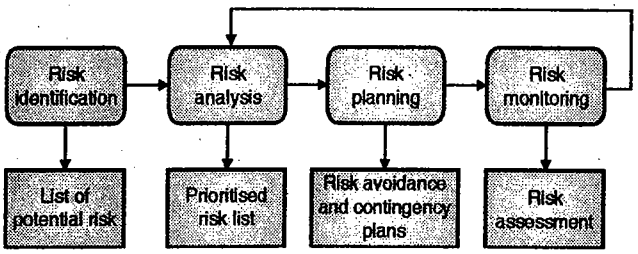


Fig. 11.1.1 : The Risks management process

11.1.1 Reasons for Project Delay

Review Question

Q. Mention the reasons for project delay.

Following are few common reasons for project delay :

- One major factor that has been identified as reasons for project delay in most projects is design errors.
- It is important to note that proper representation of client's requirement and the blue print to achieving good technical input to project execution are usually mapped out base on project designs. Thus a design with errors practically means wrong or insufficient representation of project deliverables.
- Delay in project could be as a result of scope change. Scope is the term that defines the entire deliverables that is expected at the end of a project. Therefore, logically, it can be said that all project plans, estimation, schedule, quality and base lines are usually designed base in the initial project scope. Thus, any change in the project scope during execution will mean that the entire initial project plan will have to be reviewed such that a reviewed budget, schedule and quality will have to be developed.
- Another major reason for delay in project is inappropriate and inadequate procurement and faulty contractual management system. Contracts read out virtually every aspect of a business correlation, including payment terms, pricing, and service levels. Therefore a contract that has not highlighted the entire project scenario may lead to dispute in the contract system.
- The complexity of project could also be a contributing factor to delay and cost overrun. Complexity could be defined in terms of the size of the project, most mega projects tend to have relatively long implementation period when compared to small projects.
- The post execution phase of a project contains potential factors that can lead to delays and cost overrun. Being the very last part of the project life-cycle, it is often been ignored even by organizations, especially in multi-project environments.
- Slow closeout could be seen as dragging the various handover activities course by unresolved disputes linked with client acceptance, contracts and procurement, change order issues not resolved, final change orders not issued, poor close out of final account, poor documentation of project success and lessons learnt, slow client

acceptance and failing to close the work order can allow unexpected delay.

11.2 Risk Strategies

11.2.1 Reactive Versus Proactive Risk Strategies

1. Reactive risk strategy

- In this type of risk strategy, the project is monitored closely for the likely risks that may occur.
- Resources are monitored to guess the possible risks and deal with them so that they do not become the problems for budget slippage and cost slippage.
- Generally in reactive risk strategy, the development team members are directly not involved in the risks occurring.
- And after occurrence of the risks, the development team members do some rapid action to overcome the problems.
- This rapid action is called as "fire fighting mode".
- After failure of all these actions, the concept of "crisis management" comes into picture and the project is in deep trouble and uncertainty happens.

2. Proactive risk strategy

- A proactive risk strategy is initiated when the actual technical work begins.
- The potential risks are identified and the probability and impact are assessed so that the development team members establish an appropriate plan to manage all these risk.
- The main goal to avoid the risks initially.
- But all the risks cannot be avoided, therefore the development team creates a contingency plan that will control the risks in an effective manner.

11.3 Risk Identification

- Risk identification is related to discovering possible risks to the project. It is a systematic attempt to specify threats to project plan.
- There are two types of risks :

1. Generic risk
2. Product specific risk

1. Generic risk

They are potential threat to every software project.

2. Product specific risk

- The project plan and software statement of scope are examined to find out threats due to special characteristics.
- The check list can be created of risk and then focus is subset of known and predictable risks in following subcategories :

(i) Product size
(ii) Business impact
(iii) Customer character
(iv) Process definition
(v) Development environment
(vi) Technology to the limit
(vii) Staff size and experience

(i) Product size

Risk involved in the overall size of the software.

(ii) Business impact

Risk involved in constraints imposed by management or the market.

(iii) Customer characteristics

Risk involved in the sophistication of the customer and ability of the developer to communicate the customer properly.

(iv) Process definition

Risk involved in defining the software process followed by software development organization.

(v) Development environment

Risk involved in availability and quality of the tools used in the development process.

(vi) Technology to the limit

Risk involved in complexity of system and risks associated with the new technology used.

(vii) Staff size and experience

Risk involved in overall development teams i.e. experience of developer, skill set of team members.

11.3.1 Assessing Overall Project Risk

Following are the set of questions obtained from some previous successfully completed projects :

- Are the top software managers and customer manages are committed to help the project ?
- Are the end-users are involved in the development process with enthusiasm ?
- Are all the requirements elicited properly by the customers and are these questions well understood by the development team ?
- Have the customers involved completely in the development process with development team while defining the requirements ?
- Are all the end users mentioning their realistic expectations from the project ?
- Is the project scope stable or not ?
- Are there in the development team proper blend of skilled team members ?
- Are the requirements of the project stable ?
- Are the project team members well familiar with the latest technologies ?
- Are there enough team members in a team allotted for a project ?
- Are all the customers and users agree on the requirements of the project to be built ?

If these questions are answered negatively, then the project may be at risk. In another words, we can say that the degree of risk is directly proportional the number of negative answers given.

11.3.2 Risk Components and Drivers

- In the project development process, it is essential for a project manager to identify the factors that affect the following **risk components** :
  - Performance
  - Cost
  - Support and
  - Schedule
- The factors that affect the risk components are also called as **risk drivers**. In the context of risk management, the above components can be explained as follows :
  - **Performance risk :** It is defined as the degree of uncertainty that an application satisfy the requirement expected by the customer.
  - **Cost risk :** It is defined as the degree of uncertainty that a project does not exceed its budget.
  - **Support risk :** It is defined as the degree of uncertainty that the final product will be adaptable and easy to maintain.
  - **Schedule risk :** It is defined as the degree of uncertainty that the final product will be delivered on its deadline as mentioned in the documents.
- The impact of the risk drivers on risk components are classified in four different classes as follows :
  - Negligible impact
  - Marginal impact
  - Critical impact, and
  - Catastrophic impact
- We can illustrate all these risk components (i.e. performance, cost, support and schedule) in the Table 11.3.1.

Table 11.3.1 : The table showing impact analysis

Risk component / Class of impact	Performance	Cost	Support	Schedule
Negligible impact	If performance failure occurs, then it will cause inconvenience.	Errors can cause no serious impact on the project budget	The technical performance is not compromised. It can be supported with ease.	The delivery is before the deadline.
Marginal impact	There is some degradation in technical performance.	Some schedule slips can cause the cost within limits.	Some small degradation in performance.	The delivery is within the limits.
Critical impact	There is question mark in completion and success of the project.	The failure can cause project delays and thus financial overruns.	Some delay in project delivery due to modifications in the LOC.	There is schedule slippage due to LOC delays.
Catastrophic impact (Disastrous impact)	If performance failure occurs, then simply it is the project failure.	Failure can cause drastic cost overrun and project delays.	The final product is non-responsive and can not be supported.	Unachievable deadline due to increased cost and financial crunches.

11.4 Risk Projection

- Risk projection is interchangeably called as Risk estimation also. The risk projection rates the risk in following two ways :
- The probability of risk occurrence and
- The consequences of the risk occurred.
- Following are four important risk projection activities that every manager, developer should perform :
  - Make a scale to measure the likelihood of the risk.
  - Describe the consequences of the risks.
  - Estimate its impact and
  - Write down overall accuracy of the risk projection to avoid misunderstandings.

11.4.1 Developing a Risk Table

Review Question

Q. Prepare a sample risk table and explain the RMMM plan for the same.

- A risk table gives very useful technique to project managers for the risk projection. Following Table 11.4.1 is an example of risk table :

Table 11.4.1 : A sample risk table

Risks	Category	Probability	Impact	RMMM
Size estimate may be significantly low	PS	60%	2	
Larger number of users than planned	PS	30%	3	
Less reuse then planned	PS	70%	2	
End-users resist system	BU	40%	3	
Delivery deadline will be tightened	BU	50%	2	
Funding will be lost	CU	40%	1	
Customer will change requirement	PS	80%	2	
Technology will not meet expectations	TE	30%	1	
Lack of training on tools	DE	80%	3	
Staff inexperienced	ST	30%	2	
Staff turnover will be high	ST	60%	2	

In the above table the impact values are as follows :

For the disastrous situations → 1

For some critical situations → 2

For the average situations → 3

For the negligible situations → 4

- In the above table the categories used are as follows :
  - PS used for project size risk
  - BU used for business risk
  - CU used loss of funds etc.
- The risk table is prepared right in the beginning of the development process.
- It is very much useful in analyzing the various types of risks associated. The risks are categorized in different categories as shown in Table 11.4.1.
- The probability of occurrence of particular risks is also estimated and associated impact values are listed in the above table. The project manager goes through this table and gives a cutoff line.
- All the risks that fall below the cutoff line are evaluated again. Similarly the risks that are above the cutoff line are properly managed by the project manager.

11.4.2 Assessing Risk

Following are three important factors that affect the result if risk occurs :

- **The nature of risk :** It is important to know the nature of the risk expected to occur. For example, if the hardware interface is not defined properly, then it will cause the problems during integration.
- **The scope of risk :** It looks into the severity of the risks and how the end-users will be affected by the occurrence of the risk.

- **The timing of risk :** It is important to note that for how long the impacts of risk will remain.
- Consider the following risk analysis approach. In order to determine the impact of risk, following steps are recommended :
- Find the average probability of occurrence of the risk for the risk components like performance, cost, support and schedule risk.
- Calculate the impact of each component.
- Complete the risk table and perform analysis on the result.
- Let RE is the overall risk exposure and it written as :

$$RE = P \times C$$

Where, P = Probability of occurrence of a risk

C = The cost incurred to the project, if risk occurs.

- Consider the following example to understand above equation :

Example 1 : Let a software team describes the probable risk in the following manner :

a) **Risk identification :** There are 75 % reusable components that can be integrated to the complete application. The remaining 25 % are the functionalities that are customized.

b) **Risk probability :** Let there are 85 % probable risks.

c) **Risk impact :** Let there are only 65 % reusable components planned out of 75 % used. Then in this scenario, nearly 15 components have to be developed from scratch, i.e. they are not reusable components. For calculation purpose, we consider 100 LOC on an average. Assume that $20 are incurred for each of the LOC, the total cost to develop the complete project can be computed as :

$15 \times 100 \times 20 = \$30,000.$

Thus the **risk exposure** can be computed as follows :

$$RE = 0.85 \times 30,000$$

$$= 25,500 \text{ (Approx.)}$$

In this way, the RE can be calculated for each of risk in the risk table.

Benefits using risk exposure

- The risk exposure is useful in adjusting the final cost of the project.
- It is also useful in cost estimation of the entire project.
- It is also useful in some situation where there is risk of schedule slippage. It can predict the probable risks.
- If the risk is predicted, then easily the organization can increase the manpower in that project to recover the schedule slippage.

11.4.3 Project Plan

Review Question

Q. With suitable examples, explain the differences between 'known risks' and 'predictable risks'.

- Planned and controlled software projects are conducted for one and only reason because it is the only way known to manage complexity.
- Project failure rate remains higher than it should be even though the success rate for software projects has improved. One of the reasons is that the customers lose interest quickly (because what they have requested was not really as important as they first thought), and the projects are cancelled.
- The software engineers and the software project manager must follow certain guidelines for project plans in order to avoid project failure :
- Carefully design a set of common warning signs,

- Understand the critical success factors that lead to good project management, and
- Develop a common sense approach for planning, monitoring, and controlling the project.

11.4.4 Differences between Known Risks and Predictable Risks

Sr. No.	Known risks	Predictable risks
1.	It can be uncovered after careful evaluation of project plan, business and technical environment in which the project is being developed.	The risks that are extrapolated from past project experiences are known as predictable risks.
2.	Risk associated with availability and quality of the tools to be used.	Risk associated with changing requirement from customer.
3.	Risk associated with overall technical and project experience of the software engineer who will do the work.	Risk associated with the complexity of the system to be built.
4.	e.g. unrealistic delivery date, lack of documented requirements or software scope, poor development environment.	e.g., staff turnover, poor communication with the customer, dilution of staff effort as ongoing maintenance requests are serviced.

11.5 RMMM

SPPU - Dec.12, May 14

University Questions

Q. What is risk mitigation, risk monitoring, risk management ? Explain in brief.

(Dec. 2012,10 Marks)

Q. What is RMMM? Explain in detail.

(May 2014, 9 Marks)

- Risk analysis actually helps the project development team to build a strategy to handle all possible risks. Following are three important issues (or steps) that must be considered, for developing effective strategies :
  - Risk avoidance (i.e. Risk mitigation)
  - Risk monitoring
  - Risk management and planning.
- In order to avoid the risk, the best approach is proactive approach. In the proactive approach, the development process starts with risk mitigation plan and it helps in avoiding possible risks.
- The step is risk monitoring that begins from the start of the development process. The project manager is generally responsible for keeping vigilance on the factors that can cause risk to occur.
- The project manager starts monitoring with the information received from the risk mitigation step. He scans all the documents prepared in the risk mitigation step.
- The third and final step is risk management and planning which assumes that the risk has occurred and accordingly the manager has to take necessary action.
- The Risk Mitigation, Monitoring and Management (RMMM) steps incur extra project cost.

Syllabus Topic : Planning Risk Management

11.5.1 The RMMM Plan

- Risk management strategies should be included in the project plan itself. The risk management plan is usually added as a separate plan in the project plan. This is

referred as risk mitigation, monitoring and management plan or RMMM plan.

- The RMMM plan is executed as a separate plan to evaluate the risk. Each of the risks are documented separately.
- The RMMM plan is well documented in the beginning of the project itself. Once the project begins, the mitigation and monitoring activities are started.
- The risk monitoring has main three objectives :
- Check whether the predicted risks actually occur.
- All the risk assessment steps are properly followed, and
- Collect all the necessary information that can be useful for future risk analysis.
- Following are the list of probable risks that may occur during project development process. Here in this section, we explain each of the risks mentioned below discuss them keeping RMMM Plan in mind :

1. Computer Crash
2. Late Delivery
3. Technology will not Meet Expectations
4. End Users Resist System
5. Changes in Requirements
6. Lack of Development Experience
7. Lack of Database Stability
8. Poor Quality Documentation
9. Deviation from Software Engineering Standards
10. Poor Comments in Code

(1) Risk : Computer Crash

(a) Mitigation

- The computer crash can cause the loss of important data and ultimately it will result in failure of the project.
- It is always recommended that software development organization should keep multiple copies of the data at multiple locations to avoid loss.

(b) Monitoring

The project manager must keep a watch on the working infrastructure and working environment before the actual development work starts.

(c) Management

Any inconvenience in working environment must be noticed immediately and a proper action should be taken to make the system stable.

(2) Risk : Late Delivery

(a) Mitigation

- The late delivery will result in approval from the customer. If the customer is reluctant to give the approval, then the development organization will get a bad image and it will affect the organization for getting future projects.
- So the important steps and precautionary measure are taken to timely delivery of the project.

(b) Monitoring

Continuous monitoring is required during the entire development process. The development schedule must be observed strictly and if any slippage occurs, it be noted seriously and necessary action should be taken for the recovery.

(c) Management

The delayed project can move to critical state and ultimately can cause project failure. The project manager must look into the matter seriously and should negotiate with the customer for extending the deadline as a final solution.

(3) Risk : Technology Does Not Meet Specifications

(a) Mitigation

The formal and informal meeting must be conducted with the customer to avoid this risk. It guarantees that realistic products are being developed as per customer's need.

(b) Monitoring

The proper and frequent communication is needed to understand each other and ultimately the requirement.

(c) Management

- If the idea of the project specification does not meet the requirement narrated by the customer, then the management team must take quick action to resolve this issue.
- A meeting must be conducted to understand the problems.

(4) Risk : End Users Resist System

(a) Mitigation

The application must be developed keeping the end user in mind. The user interface must be user friendly and convenient to operate.

(b) Monitoring

Any mismatch must be monitored immediately and it should be brought to development team.

(c) Management

- If the system is resisted by the user then the software must be evaluated completely and it is necessary to uncover the reasons and user interface should also be investigated properly.
- After uncovering the reasons for customer resistance, the immediate actions are required.

(5) Risk : Changes in Requirements

(a) Mitigation

To avoid the changes in requirement by the customer, it always better to keep the record of requirements given by the customer. There must be formal and informal meetings conducted between the development team and the customer.

(b) Monitoring

The meetings between the customer and the development organization must be conducted to understand each other and the requirement also.

(c) Management

To rectify any notified problem, a meeting should be conducted to discuss at the issue and the resolution of the issue.

(6) Risk : Lack of Development Experience

(a) Mitigation

The development team members must be updated and they should be well experienced to use the development tools needed.

(b) Monitoring

The team members must look into other members also to find any weak link in the chain. If such cases are observed, then it should be brought to management team to handle.

(c) Management

The experienced members must help those who are proving a weak link.

(7) Risk : Database is not Stable

(a) Mitigation

The development team should communicate the database administrator to keep it stable. All the functions, procedures and database operation must be operated error free.

(b) Monitoring

Each of the team members must monitor continuously the functioning of all database operations. If any inconvenience is reported, it must be brought o attention.

(c) Management

Any noticed problem must be resolved instantly.

(8) Risk : Poor Quality Documentation

(a) Mitigation

All the stakeholders of the project development must conduct meeting to formulate a good documentation. Any suggestions must be incorporated to enhance the quality of the documentation.

(b) Monitoring

To keep the testing and development in normal condition, the documentation must be referred time-to-time.

(c) Management

Any good opinion must be included in the documentation and any unnecessary topic should be removed.

(9) Risk : Deviation from Software Engineering Standards

(a) Mitigation

To avoid deviation from the standards, the development team must be familiar with the entire software engineering standards. They should have proper knowledge and understanding of the process.

(b) Monitoring

The technical reviews must be taken to determine any deviation.

(c) Management

If any deviation noticed, it should be addressed immediately and the management team must guide the development team to bring them on the right track.

(10) Risk : Poor Comments in Code

(a) Mitigation

A writing standard must be followed while coding. All the functions and sub-functions must be commented properly for better understanding of the code.

(b) Monitoring

The codes must be reviewed on a regular basis to determine any poor comments.

(c) Management

If any poor comments are observed, action must be taken to minimize the poor commenting and refining comments as necessary.

11.5.2 Example

Review Question

Q. Identify any two risks for your exam. Perform risk assessment and prepare the RMMM plan.

- The goal of the risk mitigation, monitoring and management plan is to identify as many potential risks as possible.
- Following is the Risk Identification Checklist for the exam :
  - Step 1 – Hazards : Possible hazards may include fire, electrical appliances, lighting, furniture, security etc.

- Step 2 – Who might be harmed ? : There is no need to list individuals by name, just think about groups of people who might be affected by those hazards, for example: candidates, teachers, examiners, stewards, escorts etc.
- What is being done to control the risk ? For the hazards listed, do the precautions already taken represent good practice and reduce risk as far as reasonably practical ? Have you provided adequate information or procedures?
- Is more action needed to control the risk ? : Where the risk is not adequately controlled, indicate what more you need to do.

Risks	Category	Probability	Impact	RMMM
Exam Hazards	EH	25%	2	
People being harmed.	PH	80 %	3	
Action taken to control the risk.	AT	50%	2	
More action needed to control the risk.	MA	20%	1	

- In the above table the impact values are as follows :
  - For the Exam Hazards → 1
  - For some critical situations → 2
  - For the average situations → 3
  - For the negligible situations → 4
- In the above table the categories used are as follows :
  - EH used for exam hazards.
  - PH used for People being harmed.
- AT used for Action taken to control the risk.

❑❑❑

UNIT VI

CHAPTER 12

Software Configuration Management

Syllabus

Software configuration management: SCM basics, SCM repository, SCM process, SCM tools such as GitHub, CASE – taxonomy, tool-kits, workbenches, environments, components of CASE, categories (upper, lower and integrated CASE tools), technology evolution, process trends, collaborative development, test-driven development, global software development challenges

Syllabus Topic : Software Configuration Management

12.1 Software Configuration Management

SPPU - May 13, May 14

University Question

Q. Write a short note on: Software Configuration Management. **(May 2013, May 2014, 5 Marks)**

- Basically the output of any software process contains the information based on various inputs. These output can be broadly divided into three important categories as follow :
  - The computer programs
  - The work products and
  - The data that consist of the information produced.

  All these information parts collectively called as software configuration.
- If each configuration item simply led to other items, little confusion would result. During the process, change takes place which may be unfortunate. But change can occur any time and for any undefined reason. The change is the only constant.
- The first law of system engineering says that "No matter where you are in the system life cycle, the system will change, and the desire to change it will persist throughout the life cycle".
- Following are four basic sources of change :
  - Every new business conditions and market conditions may cause the change in product requirement and business rules definitions.
  - Customers may require change in data presented by the information system. The customers may also demand various functionality delivered by their product of interest. The customers seek some new services. All these changes are customer oriented.
  - As the business grows or its downsizing may cause various changes in priorities of the project and also restructuring of the team happens.
  - The budget and time scheduling constraints are also important factors for change in system or product.

SCM is a set of activities used for managing the change during the life cycle of computer software. The software configuration

management can also be considered as a software quality assurance activity during the development process.

Syllabus Topic : SCM Basics

12.1.1 SCM Basics

SPPU - May 12, Dec. 14

University Questions

Q. What are elements that exist when an effective SCM system is implemented? Explain each in detail. **(May 2012, 5 Marks)**

Q. Write short note on: Elements of a Configuration management system. **(Dec. 2014, 6 Marks)**

- Four basic elements that should exist when a configuration management system is developed :
  1. **Component elements :** The tools in the file management system uses the software configuration item.
  2. **Process elements :** The process elements or the procedures uses effective approach towards the change management in engineering and use of computer software.
  3. **Construction elements :** The automated tools are used in construction or the development process and ensuring the validated components should be assembled.
  4. **Human elements :** In order to make the effective use of SCM, the team makes the use of various tools and process feature.
- These elements are not mutually exclusive. For example, component elements work in conjunction with construction elements as the software process evolves. Process elements guide many human activities that are related to SCM and might therefore be considered human elements as well.

12.1.2 Baselines

- The change is the only constant in software development life cycle. The customer want to modify the requirement as the model gets ready. Since in the beginning, even customer is not fully aware of the product requirement. As the development begins, customers need lot of changes in the requirement.
- Once customers modify their requirement, it is now manager who modifies the project strategy.
- Actually as time passes, all the people involved in the product development process come to know exact need, the approach and how it will be done in the prescribed amount of time.
- The additional knowledge is required to know the exact requirement. It is very difficult for most of the software engineers to accept this statement that most of the changes are justified.
- The baseline is SCM that help in development process without affecting much the schedule and control the changes.
- The IEEE provides a baseline as follows :
- "A specification and requirement that is agreed upon between customer and developer is a basis for the product development and these requirements can be changed only through change control procedures".

12.1.3 Software Configuration Items

- Basically SCI i.e. software configuration item is the integral part of software engineering development process. It is a part of large specification or we can say that one test case among large suite of test cases.
- In fact the SCI is a document or the program component like C++ functions or a Java applet.

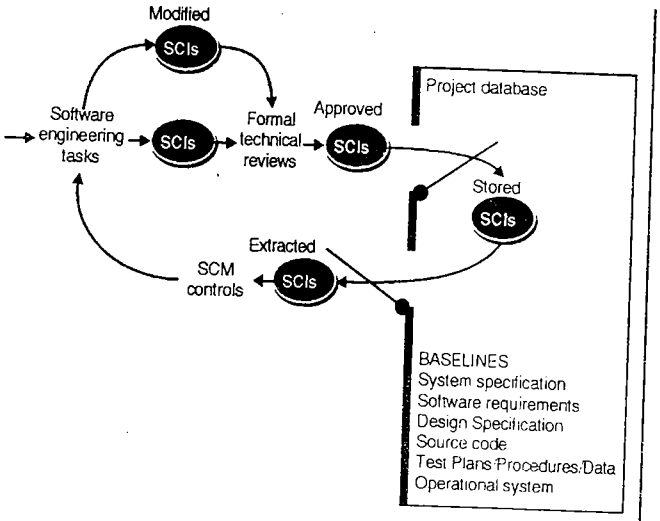


Fig. 12.1.1 : Baselined SCIs and the project database

- Most of the organizations use software tools under configuration control to help development process. In fact the editors, compilers, browsers and various automated tools are the integral part of software configuration.
- In fact the SCIs are catalogued in the project database with a single name and they form configuration objects. These objects are configuration object and it has a name, attribute and it has certain relationship with other configuration objects.

  Referring to Fig. 12.1.2, the configuration objects, Design Specification, Data Model, Component N, Source Code and Test Specification are each defined separately.

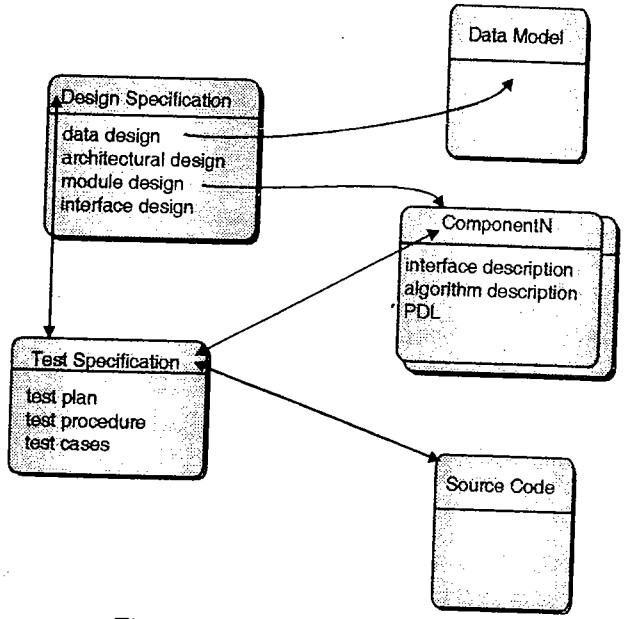


Fig. 12.1.2 : Configuration objects

Syllabus Topic : SCM Repository

12.2 SCM Repository

SPPU - May 15, Dec. 15, May 16

University Question

Q. What is software configuration management repository ?

(May 2015, Dec. 2015, May 2016, 5 Marks)

- In the early days of software engineering, software configuration items were maintained as paper documents (or punched computer cards), placed in file folders or three-ring binders, and stored in metal cabinets.
- This approach was problematic for many reasons:
  - To find a SCI is a difficult task when it is needed.
  - To determine which items were changed and by whom. It is always challenging.
  - To develop a new version from an existing program is prone to errors and time consuming too.
  - To describe detailed relationship is actually impossible.
- As discussed earlier that SCIs are catalogued in the project database with a single name, they reside in the repository. The repository is a database that stores the software engineering information. The software developer or engineer interacts with the repository by using built in tools within repository.

12.2.1 The Role of the Repository

SPPU - May 15, Dec. 15

University Question

Q. Discuss role of SCM repository.

(May 2015, Dec. 2015, 5 Marks)

The SCM repository is the set of mechanisms and data structures that allow a software team to manage change in an effective manner. The repository will perform all the fundamental operations of database management system and in addition it will perform the following operations :

- **Data integrity :** Data integrity validates all the entries to the repository and make sure that the consistency among various objects intact and takes care of all the modifications takes place. It will also ensure cascading modifications i.e. change in one object causes change in a dependent object also.
- **Information sharing :** It is mechanism for sharing information among various developers and between various tools. These tools manage and control multi-user access to data, and locks or unlock objects to retain its consistency.
- **Tool integration :** Tool integration is a data model that can be used by many software engineering tools to control access to the data, and performs appropriate configuration management functions.
- **Data integration :** Data integration provides database functions that allow various SCM tasks to be performed on one or more SCIs.
- **Document standardization :** Document standardization is an important task for defining the objects. This standardization is a good approach for making software engineering documents.
- **Methodology enforcement :** Methodology enforcement defines an (E-R model) i.e. entity-relationship model available in the databases i.e. repository. This model may be used as a process model for software engineering. It is mandatory that the relationships and objects must define before building the contents of the repository.
- To achieve these functions, the database is used as a meta-model. This meta-model exhibits the information and how this information is stored in the databases i.e. repository. This meta-model also checks data security and integrity.

12.2.2 General Features and Content

SPPU - May 15, Dec. 15

University Question

Q. Discuss features of SCM repository. **(May 2015, Dec. 2015, 5 Marks)**

- The contents of databases and features of databases are considered from two perspective :
  - What data is to be stored in the databases
  - What services are provided by the databases

A detailed breakdown of types of representations, documents, and work products that are stored in the repository is presented in Fig. 12.2.1.

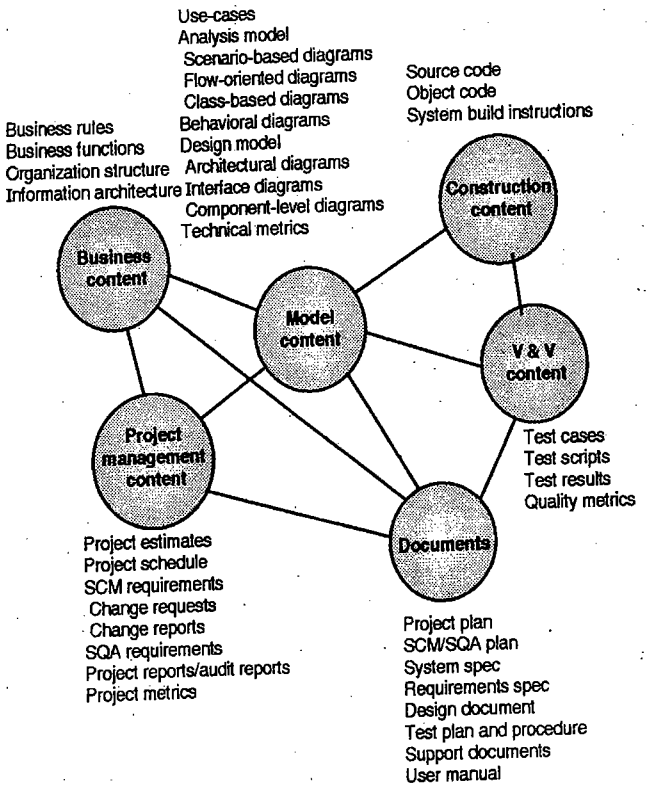


Fig. 12.2.1 : Content of the repository

- A robust repository provides two different classes of services :
  1. The same types of services that might be expected from any sophisticated database management system and
  2. Services that is specific to the software engineering environment.
- A repository that serves a software engineering team should :
  - Integrate with or directly support process management functions;
  - Support specific rules that govern the SCM function and the data maintained within the repository;
  - Provide an interface to other software engineering tools; and
  - Accommodate storage of sophisticated data objects (e.g., text, graphics, video, audio).

12.2.3 SCM Features

To support SCM, the repository must have a tool set that provides support for the following features :

1. Versioning
2. Dependency tracking and change management
3. Requirements tracing
4. Configuration management
5. Audit trails

1. Versioning

- In the development process, as the project progresses, various versions of the product will be developed and the database will save all these versions.
- The repository will keep track of these versions in order to make effective management of the product delivery or the product releases.
- The history of all releases will be used by the developers to make effective testing and debugging.

2. Dependency tracking and change management

- The databases or the repository stores various relationships among the configuration objects.
- The relationships may be between entities and processes, or between application , design and component design and between all the design elements etc.
- Some relationships are optional and some of the relationships are mandatory relationships that have various dependencies.
- So to keep the track of previous history and relationships is very important for the consistency of the information present in the databases. The new releases of the final product are also dependent on the history stored in the repository. This will be useful in the improvement process.

3. Requirements tracing

- The requirement tracing will provide the ability to trace all the design components and releases. This tracing results from a specific requirement called as forward tracing.
- It will also useful in finding that which requirements are fulfilled properly from the ready product. This is called as backward tracing.

4. Configuration management

- The configuration management is a facility to keep the track of various configurations under development.
- The series of configurations is used as the project milestones and future releases.

5. Audit trails

- The audit trails keeps additional information (i.e. meta-data) like the changes made by whom and when. Also it stores the reasons why changes have been made.
- The source of each change in the development must be stored in the repository.

12.3 SCM Process SPPU - Dec.12, May 16

University Questions

Q. What is the objective of SCM ? **(Dec. 2012, 4 Marks)**

Q. Explain SCM process in detail. **(May 2016, 4 Marks)**

- The SCM (Software Configuration Management) process consists of series of task to monitor the control on changes being occurred. The main objectives of these tasks are as follows :
  1. To identify all individual items that can define software configuration collectively.
  2. Manage the changes taking place in various individual items.
  3. To handle different versions or releases of product.
  4. To maintain the quality of the software under construction over the period of time.
- A process that achieves these objectives must be characterized in a manner that enables a software team to develop answers to a set of complex questions:
  - How does a software team identify the discrete elements of a software configuration?
  - How an organization manages the changes being done in existing release or the existing version? The modification should be incorporated efficiently.
  - How an organization keeps the track and control of new releases?
  - In an organization, who is responsible for authorizing all these changes?
  - The mechanism used to let others know the changes taking place and implemented?
- These questions lead us to the definition of five SCM tasks - identification, version control and change control, configuration auditing, and reporting, as illustrated in Fig. 12.3.1.

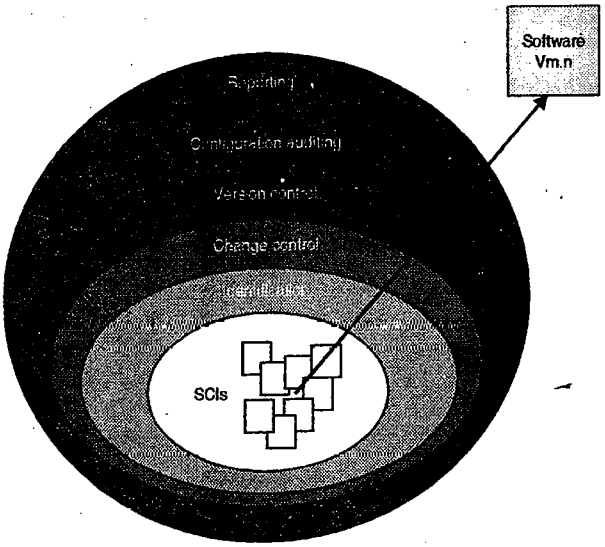


Fig. 12.3.1 : Layers of the SCM process

12.3.1 Identification of Objects in the Software Configuration

- To control and manage software configuration items, each should be separately named and then organized using an object-oriented approach. Two types of objects can be identified :
  1. Basic objects and
  2. Aggregate objects.

1. **Basic object :** A basic object is a unit of information that has been created by a software engineer during analysis, design, code, or test.

2. **Aggregate object :** An aggregate object is a collection of basic objects and other aggregate objects.

- Since each of the object in the product has some distinct features that make the object different from other objects. The object has a unique name, description and list of resources associated with it. The name of the object should be very clear and distinct.
- The description of the object should identify the type of software configuration item i.e. SCI represented by that object.

12.3.2 Version Control

- The version control actually controls the new releases or new versions. It combines the procedures and tools in order to control various versions of configuration objects.
- Any version control management system has four major capabilities that is integrated in the version control system itself :
  1. The repository will store all the related configuration objects.
  2. The repository will store all the versions of configuration objects.
  3. It has a facility to provide the related information about configuration objects so that a software engineer will construct a new version based on those information.
  4. To track all the issues in development process by using a special tracking facility in the version control.

12.3.3 Change Control SPPU - May 12, May 14

University Question

Q. Write short note on: Change control process.
(May 2012, May 2014, 5 Marks)

- In a development of a larger software system, the changes may be uncontrolled and it leads to a complex situation. In such projects the change control is done partially by human and partially by some automated tools. In such a complex situation human intervention is very much necessary.
- The change control process is explained in the following Fig. 12.3.2.
- The change request is first submitted and then evaluated by a technical support staff by taking into consideration its potential side effects and the overall impact on other objects in the product. The other parameters to be evaluated are system functions, the cost of project etc.
- Based on the result of the evaluation treated as a change report, the implementation is taken into consideration. This report is submitted by change control authority i.e. CCA. The CCA is a person or a group who has the final authority to take decision on any changes and their priority.
- After approval from CCA, a change order called ECO (engineering change order) is generated for each of the change.
- The object to be changed can be placed in a directory that is controlled solely by the software engineer making the change.
- These version control mechanisms, integrated within the change control process, implement two important elements of change management :
  - o access control and
  - o Synchronization control.
- Before an SCI become a baseline, the changes should be applied. The developer will look after whether the changes are justified or not by project. The technical requirement must check properly.
- After approval from CCA, a baseline may be created and change control is implemented.

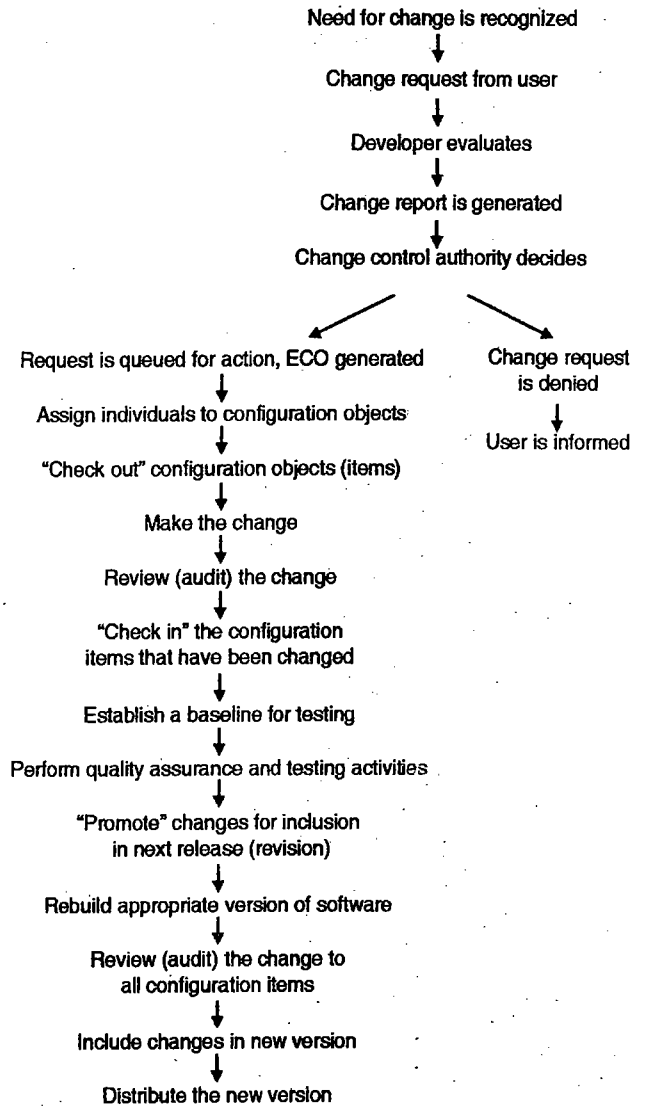


Fig. 12.3.2 : The change control process

- Once the final product is released, the formal changes must be instituted as per the figure 12.3.2. This formal change is outlined.
- The CCA plays an active role in second and third layers of change control based on size of the project.

12.3.4 Configuration Audit

- A software configuration audit addressees the following questions:
  - Whether the change mentioned in the ECO applied or not ? Incorporated any additional modifications ?
  - Whether the formal technical review conducted or not to assess technical correctness ?
  - Whether the software process followed or not and software engineering standards properly applied ?
  - Whether the changes are "highlighted" in the SCI ?
  - Whether SCM procedures for noting the change, recording it, and reporting it been followed or not ?
  - Whether all SCIs properly updated ?
- In some of the cases, the configuration audit questions are asked as part of a FTR (formal technical review). Still SCM is a formal activity. The SCM audit is conducted separately. These activities are performed by the quality assurance group.

12.3.5 Status Reporting

- Configuration status reporting is a SCM task that has following questions to answer :
  - What had happened? (Specify the changes made).
  - Who did it? (Specify the responsible person or authority approving the changes).
  - When did it happen ? (Specify the time of occurrence of the change)
  - Anything else is affected based on the changes made?
- The CSR report is generated on regular basis to keep the developers aware of the changes made and the history of the changes made. It is very much useful in new releases or constructing new versions.

12.4 SCM tools such as GitHub

- The most popular software configuration management tool is Git. Git is a distributed version control tool that was invented by Linus Torvalds to support the development of Linux. Linus named the version control tool Git which is an old English word.

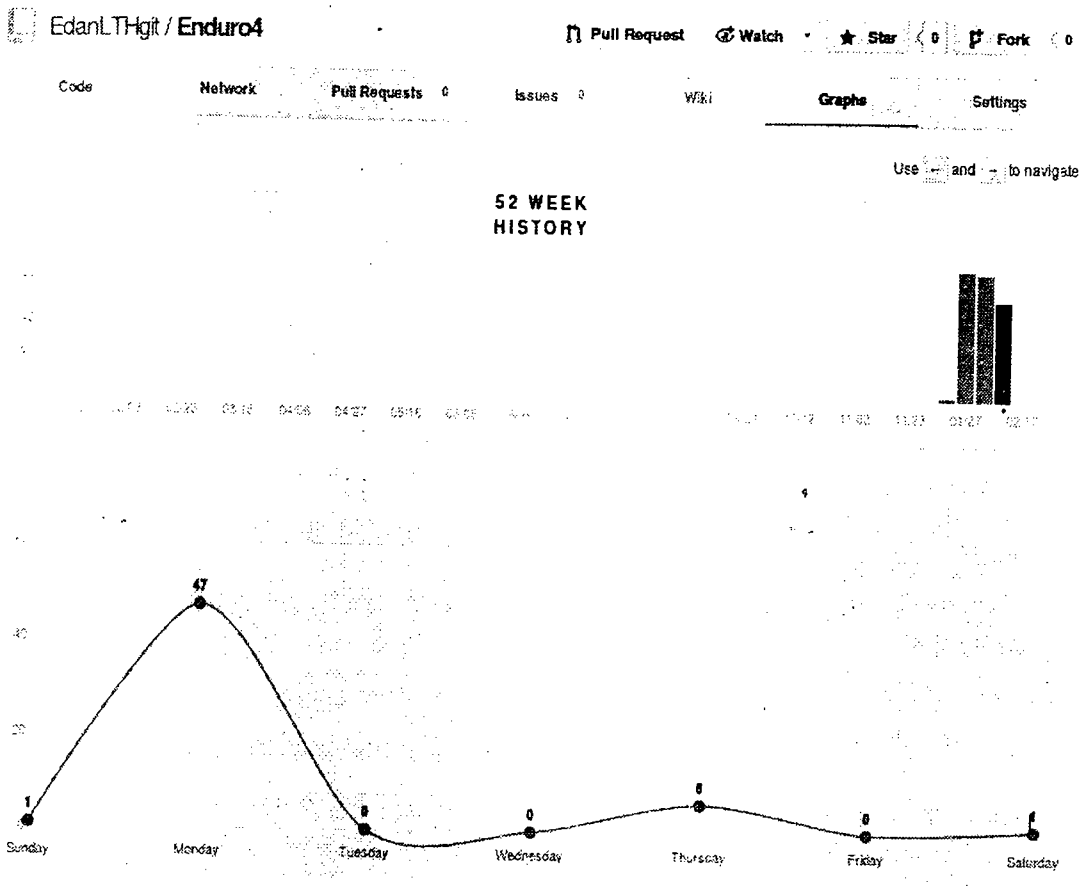


Fig. 12.4.1 : A graph produced by Github showing how code only gets committed

- The purpose of Git is to provide a reliable and versatile version control and configuration management, it does this with a little different approach than some other version control tools. Git also enables people to work together and empowers teamwork.
- Github is a Git hosting service where a person can make his own repository accessible for the public. Github is free unless you want your code to be private. It also offers a lot of socially focused features.
- Github supports Git and makes it more accessible to the public. There are a lot of code hosting services on the internet for all kinds of SCM tools if you do not want to trouble yourself with setting up servers and configuring the tools.
- Github is very focused on the collaboration between users. Each project on Github get its own wiki-page where people active in the project can document various information. Github also tracks commits and makes statistics accessible via graphs. As a user you can follow other users and enter projects.
- This is surely part of why Github is so successful, its features gives clear feedback on projects progress and allows for easy communication between developers. Many projects on Github are only used to store code and provide backup for single person projects.

12.5 Computer-Aided Software Engineering

- Computer-aided software engineering (CASE) tools are developed to help software engineers, managers and all the software practitioners in all the software activities related to the software development process. The CASE tools are actually built to automate software management activities. It also assists a software

engineer in requirement analysis, design, coding and testing, debugging.

- Software engineering is considered to be very difficult. The CASE tools reduce the amount of effort required to develop a product. In most of the project, these tools play very important role to achieve the benefits. In addition to these benefits, tools also provide different way of looking at software engineering information that improves the software quality.
- CASE tools help a software engineer or a practitioner in developing high-quality products. It also produces additional customized work due to these automation tools. Without CASE tools the thing might have been too tough.
- Computer-aided software engineering provides a platform where a software engineer automates all the manual activities. This improves the insight of engineering to produce better products. Thus CASE ensures the quality before the actual product is built.

CASE tools

CASE tools are class of software which is useful to automate the different activities in life cycle. They are useful in different software engineering phases. They are useful in requirement analysis, design, coding, testing, etc. Different CASE tools are given below.

- Business process engineering tools
- Project planning tools
- Risk analysis tools
- Project management tools
- Requirement tracing tools
- Documentation tools
- System software tools
- Quality assurance tools
- Database management tools
- Prototyping tools
- Programming tools
- Web development tools
- Static analysis tools
- Dynamic analysis tools
- Test management tools
- Client server testing tools
- Re-engineering tools

Functions of CASE tools

- Analysis
- Design
- Code generation
- Documentation

Types of CASE Tools

The general types of CASE tools are listed below:

- Diagramming tools
- Computer display and report generators
- Analysis tools
- Central repository
- Documentation Generators
- Code generators

CASE - taxonomy

Different terms involved in the CASE tools are as follows. These tools are useful to understand the CASE in details.

Workbenches

- Workbench integrates different CASE tools in one application.
- It is useful to support certain process activity.
- They generally have homogeneous and consistent interface
- They have easy invocation of tools and tools chains.
- CASE workbenches can be classified into following categories.
  - Business planning and modeling.
  - Analysis and design
  - User interface development
  - Programming

- Verification and validation
- Maintenance and reverse engineering
- Configuration management
- Project management
- Design management.

Tool-kits

- It is collection of products which are loosely integrated.
- Support provided by tool-kits is limited to programming, configuration management, project management.
- It is extended from basic set of operating system tools.

Environments

- It is nothing but the collection of CASE tools and workbenches.
- It is useful in supporting software process.
- It is classified into different categories based on basis of integration.
  - Toolkits
  - Language centered
  - Integrated
  - Fourth generation
  - Process centered

Components of CASE

- Components of CASE tools is shown in Fig. 12.5.1.

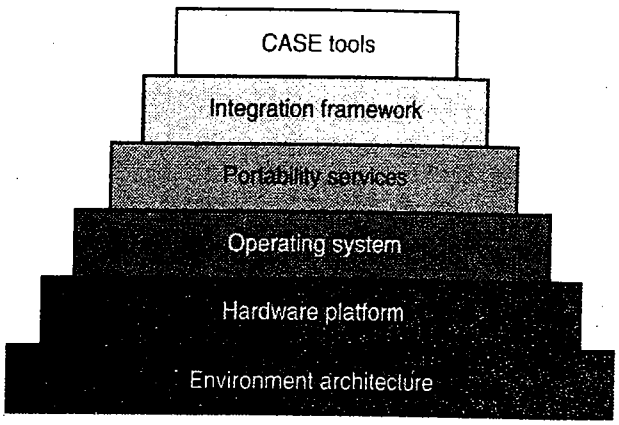


Fig. 12.5.1 : Components of CASE Tools

- It consists of different blocks like environment architecture, hardware platform, operating system, portability services, integration framework and lastly CASE tools.

Categories (upper, lower and integrated CASE tools)

- It is used in wide variety of software development life cycle methods.
- It is also useful in project identification and selection.
- It is useful in project initiation, planning and design.
- Components of CASE tools are classified into 3 main categories which are as follows:
  - Upper CASE tools
  - Lower CASE tools
  - Integrated CASE tools
- Upper CASE tools
  - These are the tools which supports software development from implementation.
  - It mainly focuses on analysis phase.
  - It also focus on design phase.
  - It includes diagramming tools, report and form generators and analysis tools.
- Lower CASE tools
  - It is useful in supporting implementation and integration tasks.
  - It is useful in database schema generation, program generation, implementation, testing, configuration, etc.
- Integrated CASE tools
  - It is also called as ICASE.
  - It supports both upper CASE tools and lower CASE tools.
  - If we consider the example of designing the form and building the database to support it at the same time.
  - Different tools are available for creating diagrams, forms, and reports.
  - It is also supporting analysis, reporting, code generation, etc.

12.6 Emerging Software Engineering Trends

- It is style of software development which is focused on public availability and communication.
- Usually communication happens using internet.
- It is used in freeware, open source software and commons based peer production.
- It is also used in agile development model.
- It is generally used in development of free software.
- It is very compatible with free software.
- They meets online for software development.
- It is evolution of integrated development environment.
- Typical functionalities are as follows :
  - o Version control system
  - o Bug tracking system
  - o Todo list
  - o Mailing list
  - o Document management system
  - o Forum
- They also involve users in the collaborative development.
- It is used in most technological disciplines.

Model-driven development

- It is software design approach for development of software system.
- It provides guidelines for structuring of specifications.
- It is kind of domain engineering.
- It is launched by object management group.
- It defines system functionalities using platform independent model.
- Related standards are as follows.
  - o Unified modeling language (UML)
  - o Meta object factory (MOF)
  - o XML metadata interchange (XMI)
  - o Enterprise distributed object computing (EDOC)
  - o Software process engineering metamodel (SPEM)
- Different tools are used in model driven architectures.
  - o Creation tools
  - o Analysis tools
  - o Transformation tools
  - o Composition tools
  - o Test tool
  - o Simulation tools
  - o Metadata management tools
  - o Reverse engineering tools
- Model driven development is shown in Fig. 12.6.1.

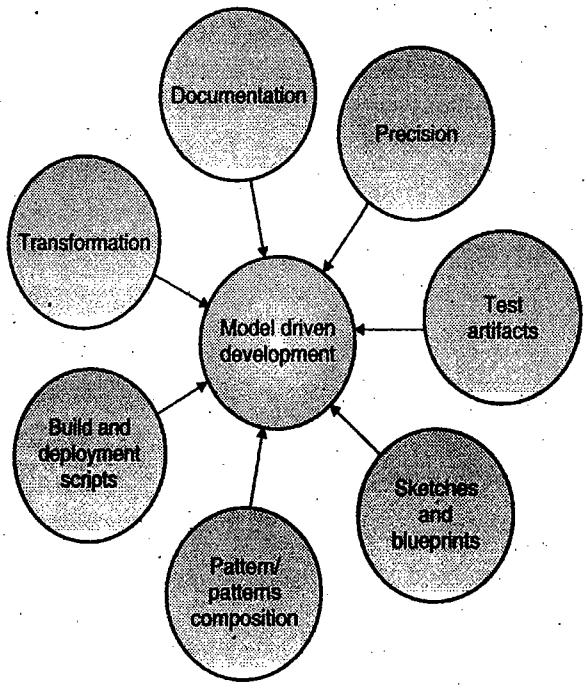


Fig. 12.6.1 : Model driven architecture

Test-driven development

- It is software development process which relies on repetition of very short development cycle.
- First, developer writes test cases which define a desired improvement.
- It is related to test first programming concept of extreme programming.
- Test driven development cycle is shown in Fig. 12.6.2
  1. Add test
  2. Run all tests
  3. Write some code
  4. Run tests
  5. Refactor code
  6. Repeat the process.

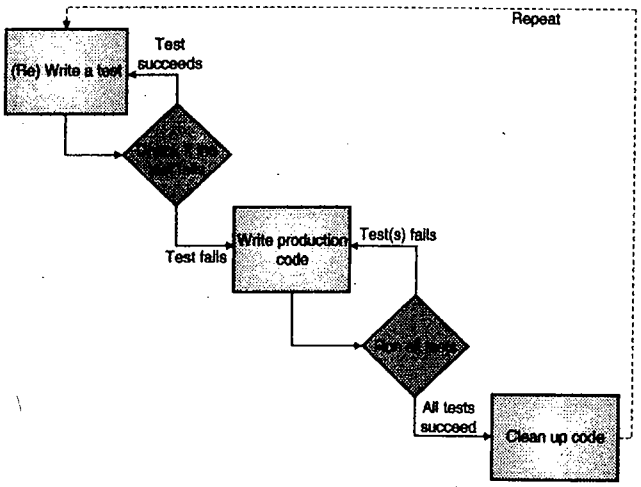


Fig. 12.6.2 : Test Driven development

Challenges of global software development

- Communication breakdown
- Coordination breakdown
- Control breakdown
- Cohesion barriers
- Culture clash
- Ability to gain market share.
- Greater flexibility and variable staffing model.
- Lower cost labor.
- Access to broader set of skilled workers.
- Ability to leverage outsourcing providers with specialized skill.
- Possibility of establishing a presence in geographies that may become new market for product.
- Ability to gain competitive edge.
- Business driver and global delivery challenges are shown in Fig. 12.6.3.

Business drivers and global delivery challenges

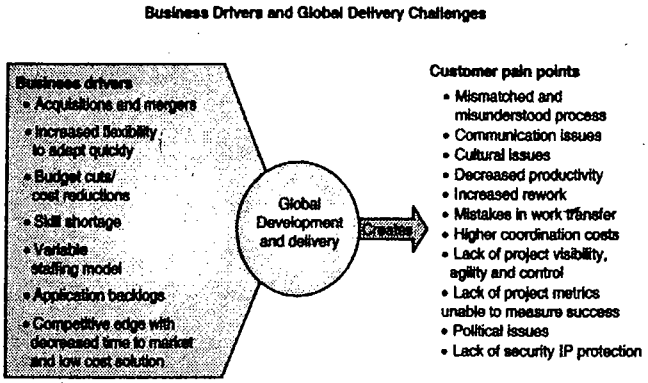


Fig. 12.6.3 : Business Driver and global delivery challenges

- Misunderstood processes or mismatched processes.
- Communication issues can lead to misunderstanding, omissions, errors and rework.

□□□

UNIT VI

CHAPTER 13

Project Management Trends

Syllabus

Emerging software engineering trends: Project Management trends: CRM, ERP: Basic concepts, Advantages and limitations, SAP, Business process reengineering, International Project Management, Case studies.

Syllabus Topic : Project Management Trends

13.1 Project Management Trends

- From a practical perspective, the two standard objectives in project management are defined to be completion of the project on time and on budget. Yet, many projects fail to meet these two criteria, despite detailed planning before execution begins and the use of modern project management software.
- Further, the failure rate of projects is higher in many modern applications than in traditional ones, due to less reliable data and the more challenging characteristics. Indeed, it can be said that, despite its recent massive growth in use, project management is a difficult to manage business process. As we discuss, this is creating extremely interesting research opportunities.
- Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) are the project management tools that are used to increase the overall profitability of a business.
- These systems overlap in some areas, and can be completely integrated in others. However, as their core functionalities are completely different, it's best for a business to first look at them as separate, stand-alone systems. When viewed separately, it's easier to see how ERP and CRM each play a role in improving efficiency and increasing sales.
- ERP and CRM are two important technology acronyms that businesses need to know :

13.1.1 CRM (Customer Relationship Management)

- CRM at its simplest is systems and processes for managing a company's interactions with current and potential customers. When we talk about CRM we usually are talking about CRM Software.
- CRM software is used to organize, automate and synchronize sales, marketing and customer service. CRM has developed to include all areas of the customer experience, keeping the customer happy and in turn keeping them loyal and more valuable to your business.

- CRM is the process of identifying potential leads/prospects, nurturing them and guiding them through the sales process to close the business. Once they are a customer it is ensuring that you maintain that relationship and encourage repeat business, either more frequent orders or higher value.
- Many aspects of CRM rely heavily on technology. CRM software will collect, manage and link information about the customer. You can use CRM software to create marketing campaigns, view a customer's entire of history of interactions with your business and use it to streamline daily business and sales tasks.

13.1.1.1 Advantages of CRM

- The business learns what the needs of its customers are and can organize their operations accordingly. The goal is to provide the products/services they expect and maximize effectiveness and efficiency of the operations.
- It allows organizations to see how customers' needs evolve overtime and adjust their operations accordingly.
- CRM can increase customer retention, favour repeated purchases, increase profit and improve the quality of the products/services provided.
- It supports planning and control activities, since they are a way of reconciling supply and demand (Ibid.). In this respect, CRM can be used to analyze current trends, forecast future trends and monitor the evolution of the demand overtime.
- CRM can help creating unique and exclusive relationships with single customers or group of customers. This have a positive, psychological effect on customers who feel considered and listened to.

13.1.1.2 Disadvantages of CRM

- It seems to be mainly focused on retention of existing customers rather than on the acquisition of new ones.
- Legal aspects (e.g.: privacy) and ethical issues should be considered during its implementation.
- It may lead organizations to discriminate group of customers. More profitable customers may enjoy better treatments and conditions than occasional customers. This may damage the image of the company.

13.1.2 ERP (Enterprise Resource Planning)

- Where CRM manages the customer, ERP is used to manage the business. ERP is a system for improving the efficiency of business processes. Like CRM, ERP allows for the rapid sharing of standardized information throughout all departments. Employees all enter information into the ERP system, creating a real-time, enterprise-wide snapshot.
- Problems in any area will automatically create alerts in other affected areas. This allows departments to begin planning for issues before they become a problem in that department. In short, by allowing the business to focus on the data, instead of the operations, ERP provides a method for streamlining business processes across the board. Microsoft Dynamics ERP can directly integrate with Dynamics CRM.
- ERP software is used to manage the business. It integrates all facets of an operation, including product planning, development, manufacturing processes, human resources, financials and sales and marketing.

13.1.2.1 Advantages of ERP

- Better organizational control, especially in large companies, where the volume of information is more than in a small company.
- Duplication of information is avoided.
- Improved communication, both internally and externally.
- Company profitability analysis can be carried out to analyze where costs are higher and where there are more sales.
- Improved decision-making process within the company.
- The company is able to react better to any unforeseen problem or situation.
- Better use of time.

13.1.2.2 Disadvantages of ERP

- The high cost of implementation and maintenance. (High initial investment)
- Adaptation to the hardware in the company.
- It is necessary to train all employees in the company so that the system is used efficiently. This is a cost for the company as well as the time and effort needed for it.
- Integration with other applications in the enterprise needed.
- Inflexibility of the system, because this is a generic system.
- There are few experts in this system.
- If the system is not applied correctly, it can be very detrimental to the company.

Syllabus Topic : SAP

13.2 SAP

- A German software company whose products allow businesses to track customer and business interactions.
- SAP is especially well-known for its Enterprise Resource Planning (ERP) and data management programs. SAP is an acronym for Systems, Applications and Products.
- SAP ERP was built based on the former SAP R/3 software. SAP R/3 consisted of various applications on top of SAP Basis, SAP's set of middleware programs and tools. All applications were built on top of the SAP Web Application Server. Extension sets were used to deliver new features and keep the core as stable as possible. The Web Application Server contained all the capabilities of SAP Basis.
- SAP ERP consists of several modules, including :
  - Financial Accounting (FI)
  - Controlling (CO)
  - Asset Accounting (AA)
  - Sales and Distribution (SD)
  - Material Management (MM)
  - Product Planning (PP)
  - Quality Management (QM)
  - Project System (PS), Plant Maintenance (PM)
  - Human Resources (HR)
- SAP ERP collects and combines data from the separate modules to provide the company or organization with enterprise resource planning.

13.2.1 SAP ERP advantages

- Allows easier global integration (barriers of currency exchange rates, language, and culture can be bridged automatically).
- Updates only need to be done once to be implemented company-wide.

- Provides real-time information, reducing the possibility of redundancy errors.
- May create a more efficient work environment for employees.
- Vendors have past knowledge and expertise on how to best build and implement a system.
- User interface is completely customizable allowing end users to dictate the operational structure of the product.

13.2.1 SAP ERP disadvantages

- Locked into relationship by contract and manageability with vendor : a contract can hold a company to the vendor until it expires and it can be unprofitable to switch vendors if switching costs are too high.
- Inflexibility : vendor packages may not fit a company's business model well and customization can be expensive.
- Return on Investment may take too long to be profitable.
- Implementations have a risk of project failure.

Syllabus Topic : Business Process Reengineering

13.3 Business Process Reengineering

- Business process re-engineering (BPR) is a business management strategy focusing on the analysis and design of workflows and business processes within an organization. BPR aimed to help organizations fundamentally rethink how they do their work in order to dramatically improve customer service, cut operational costs, and become world-class competitors.
- PR seeks to help companies radically restructure their organizations by focusing on the ground-up design of their business processes.
- A business process is a set of logically related tasks performed to achieve a defined business outcome. Re-engineering emphasized a holistic focus on business objectives and how processes related to them, encouraging full-scale recreation of processes rather than iterative optimization of sub-processes.
- Business Process Re-engineering (BPR) is an approach that recommends going through business processes to reengineer them and improve them. To reduce useless steps that might be in the process that does not add any value or benefit to the process.

13.3.1 How BPR plays a critical role in ERP implementation?

- Processes, organization, structure and information technologies are the key components of BPR, which automates business processes across the enterprise and provides an organization with a well-designed and well-managed information system. While implementing ERP, the organizations have two options to consider. Either the organization must reengineer business processes before implementing ERP or directly implement ERP and avoid reengineering.
- In the first option of reengineering business processes, before implementing ERP, the organization needs to analyze current processes, identify non-value adding activities and redesign the process to create value for the customer, and then develop in-house applications or modify an ERP system package to suit the organizations requirements.

- In this case, employees will develop a good sense of process orientation and ownership.
- This would also be a customized solution keeping with line of the organization's structure, culture, existing IT resources, employee needs and disruption to routine work during the change programmer likely to be the least. It could have a high probability of implementation.
- The drawback of this option is that the reengineered process may not be the best in the class, as the organization may not have access to world-class release and best practices. Moreover, this may be the only chance to radically improve in the near future and most attention should be paid while choosing the right ERP. Also, developing an in-house application or implementing a modified ERP is not advisable.
- The second option of implementing ERP package is to adopt ERP with minimum deviation from the standard settings. All the processes in a company should conform to the ERP model and the organization has to change its current work practices and switch over to what the ERP system offers. This approach of implementation offers a world-class efficient and effective process with built-in measures and controls, and is likely to be quickly installed.
- But if the employees do not have good understanding of their internal customer needs or current processes, or if these processes are not well defined and documented, then it is quite possible that while selecting the standard process from the ERP package, employees may not be able to perceive the difficulties likely to be encountered during the implementation stage.
- Employees would lack process ownership and orientation. Other than technical issues, issues like organization structure, culture, lack of involvement of people etc. can lead to major implementation difficulties, and full benefits of standard ERP package may not be achieved. It may lead to a situation where the organization may have to again reengineer its processes. This could be a very costly mistake.
- There is also a third option of reengineering business process during implementation of ERP. But it does not considered to be a practical option and is likely to cause maximum disruption to existing work. It should not be forgotten that during BPR and ERP initiatives, routine work is still to be carried out and customers need to be served.

Syllabus Topic : International Project Management

13.4 International Project Management

- International project management is the management of projects internationally or across borders and cultures, therefore international project management requires a specific set of skills to ensure success when managing international projects.
- International project management is becoming increasingly important in today's global business world where businesses are continuing to expand into new countries and markets, either to increase their market share or to reduce costs by utilizing more efficient resources found in other countries. International project management requires unique tools and techniques to give international projects greater chances of success.

13.4.1 International Project Management Challenges

- Project management is challenging at the best of times and managing international projects only compounds these complexities and increases these challenges, some of the challenges and global issues of international project management that will need to be considered will differ greatly depending on the countries your project will involve.
- **Differing Standards :** The problem with standards in a global sense and very much so in international project management is that they are not standard... Standards differ from country to country and consideration will need to be given as to which standards will be used will multiple standards be required. Some key areas of consideration are :
  - **Political and Legal Systems**
  - **Accounting Standards**
  - **Quality Standards and Unit of Measure**
  - **Language Barriers**
  - **Time Zone Changes**
  - **Economic Conditions**
  - **Cultural Differences**
- Of these international project management challenges, one of the most difficult to provide a framework of understanding is the cultural differences of each country, while some countries are well aligned, others can be completely different in their cultural and social norms, an obvious example of this is the differences between western countries like Australia or America and Asian Countries such as China and Japan.
- These differences can greatly impact international projects and require a specific international project management approach to solve these challenges of international projects.

❑❑❑

Note