

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

INTRODUCTION TO SOFTWARE ENGINEERING

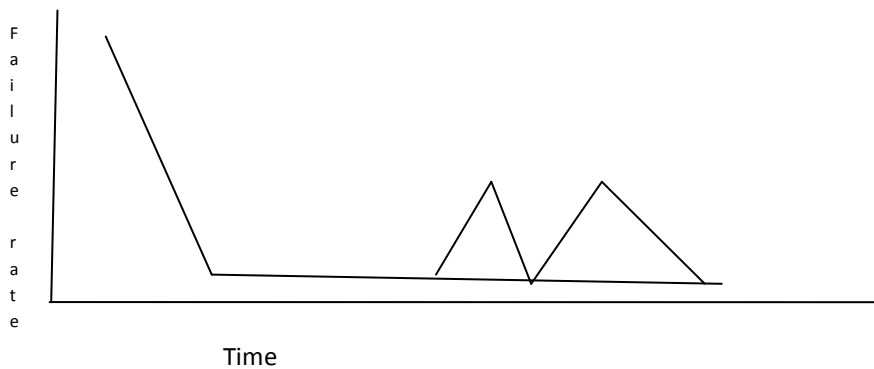
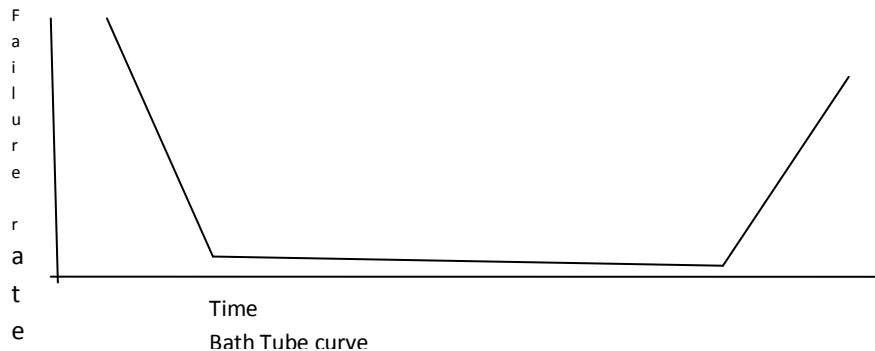
A program is a pair of software and it is a set of instruction performing a specified task. Program is developed by individuals for their personal use. It is small in size

- What is software?

Software is an instruction that when executed provide desired features, functions and performance. Or it is data structure that enables to adequately manipulate the information.

CHARACTERISTICS OF SOFTWARE:

1. Software is developed not built.
2. Software does not wear out but deteriorate.
3. Reusability of components.
4. Flexibility of adding, removing etc.

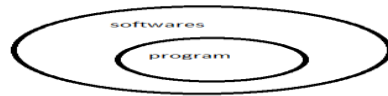


DIFFERENCE BETWEEN PROGRAMME & SOFTWARE:

| | |
|------------------------------------------------------|-------------------------------------------------------------------------------------|
| Program | software |
| A program is a set of assignment Perform some tasks. | It is an instruction that well executed & provides desire features & functionality. |
| Author of the program himself used maintain. | & Provides more functionality & flexibility |
| size of program is small | Size of software is large |
| is lack of good interface | good interface |

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE



TYPES OF SOFTWARE:

1. **SYSTEM SOFTWARE:** It is a collection of program written to service other program.
Ex: Operating system, compiler etc.
2. **APPLICATION SOFTWARE:** It consists of stand-alone program those services to Business purpose.
Ex: MS Office
3. **ENGINEERING SOFTWARE / SPECIFIC SOFTWARE:** These are algorithm, engineering & scientific software.
Ex: CAD, Simulators
4. **EMBADED SOFTWARE:** These are resides of another product & used to implement and control some features and functionality for an user.
Ex: Remote control car
5. **WEB APPLICATION SOFTWARE:** It is a set of linked http that presents information using text & limited graphics.
Ex: Computer graphics & multi media.
6. **ARTIFITIAL INTELLIGENCE SOFTWARE:** These software is used the non numerical algorithms to solve the complex problem that are not completed or computed.
Ex: Robotics & pattern reorganization.
7. **PRODUCT LINE SOFTWARE:** These software's are designed to provide a specific capability by means of customer. Ex: Whatsapp, Wechat, Viber, Line etc.
8. **REAL TIME SOFTWARE:** These are applicable to provide the information in a particular time. Ex: Time bomb.

SOFTWARE ENGINEERING:

- Software engineering is diverted into the presentation of concepts, tools & techniques during the various phases of software development.
- Software engineering is developed by Dr. Richard Thayer in 2003

SOFTWARE PROCESS:

- It is a series of predictable step or different steps/road map that helps to create a timely high quality result.
- It provides a stable controlled & organized activity that can perform by an organization.

Step 1: Information gathering/requirement analysis:-

It involves the communication or collaboration of customer/end user/stake holder. Accompany the requirement by gathering information or similar process.

Step 2: Planning /specification:-

To establish a plan for software engineering, which describes the technical task, to be conducted?

Step 3: Modeling/designing:-

The gathered information is converted in to some structured analysis or graphical representations like DFD use case activity.

Step 4: Coding/construction/implementation:-

The major part of software process is coding. Here the modifications of errors are done if required and check the errors.

Step 5: Deployment:-

After testing the software it will hand over to the stake holder.

S/w Life Cycle model

A software life cycle is series of identifiable stages i.e. software products undergoes during its life time. The life cycle model represents all activities such as requirements analysis, designing and coding, testing implementation and maintenance

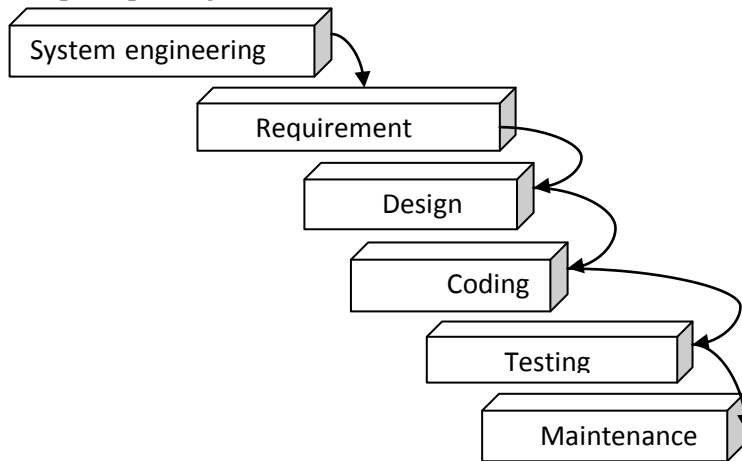
Types of model

1. Classical waterfall model

The waterfall model was proposed by Winston Royce in 1970. In the original model the phases were iterative. In practice however, it becomes rigidly sequential, therefore, came to be known as the linear sequential model.

The following figure depicts the waterfall model with iterative phases.

The principle stages of waterfall model are:



(Classical waterfall model)

System engineering:-

The software product is a part of large system. Therefore requirements are determined for all the system components and a part of these requirements are allocated for the software. This system view is needed when the software must interface with other elements like hardware, people and database.

Requirement analysis:-

Requirements are analyzed and made out before proceeding to the other process. Logical representation of the requirements analysis is required to avoid ambiguity in the requirements. This phase exactly tells the requirements and needs of the project.

This is very important and critical phase in waterfall model. This purpose of a requirements analysis is to identify the qualities required of the application, in terms of functionally, performance, ease of use, portability and so on.

This phase produces a large documents, contains a description of what the system will do without describing how it will be done. The resultant document is known as software requirement specification (SRS) document.

An SRS document must contain following:

- Detail statements of problem.

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

- Possible alternate solutions to problem.
- Functional requirements of the software system.
- Contains on the software system.

Design phase:-

The detail of the design phase is to transform the requirements specified in the SRS document into a structure that is suitable for implementation in some programming language. In technical terms, during the design phase the software architecture is derived from the SRS document.

Two differently design approaches are available: i.e.

1. Traditional design approach
2. Object-oriented design approach

Coding:-

Coding is the phase in which we actually write programs under a suitable programming language environment. It was only recognized development phase in early development processes, but it is one of several phases in a waterfall model. The output of this phase is an implemented and tested collection of modules.

Coding can be subjected to companywide standards, which may define the entire layout of programs, such as the headers for comments in every unit, naming convention for variables and sub programs.

Testing:-

During the testing phase, the modules are integrated in a planned manner. The different modules making up a software product are almost never integrated in one shot. Testing is carried out by a number of steps, during each step the system is tested and a set of previously planned modules are added to it.

The objective of system testing is to determine whether the software system performs as per requirements mentioned in SRS document. This testing is known as system testing.

The system testing is done in three phases called “alpha”, “beta” and “acceptance testing”.

Maintenance:-

Maintenance is defined as the set of activities that are performed after the system is delivered to the customer. Maintenance consists of correcting any remaining error in the systems, adaptive the applications to changes in the environment, and improving, changing or adding features and qualities the application.

Advantages:-

- It enables maximum ordering in the process implementation.
- It provides a structured template for software engineering.

Disadvantages:-

- It is difficult for the customers to give all the requirements at one go, but this is a necessity for this model.
- It is difficult for the user to anticipate whether the final system constructed according to the specifications will eventually meet their requirements.
- Customers need to have patience for working with this model.

ITERATIVE WATERFALL MODEL:

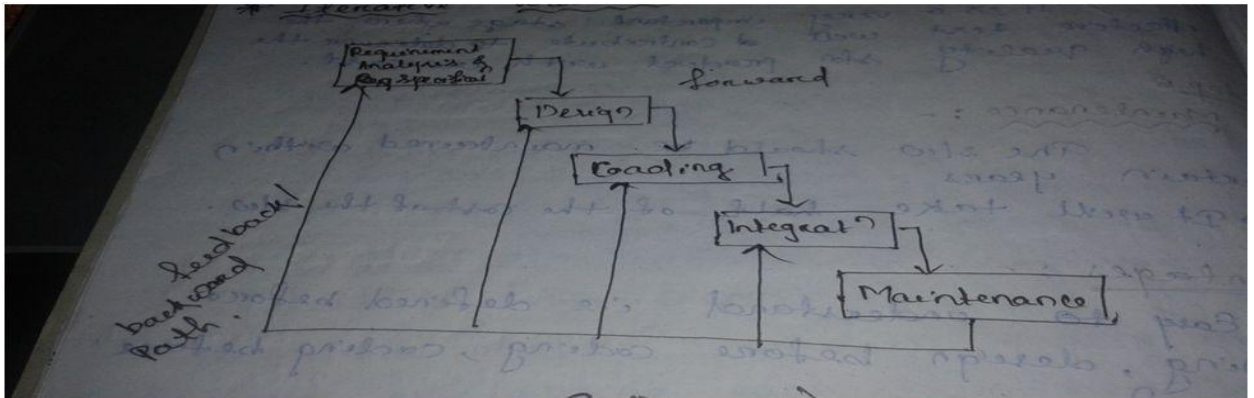
- ✓ It suggests a semantic, sequential approach to software development that begins with customer's satisfaction of requirements and progresses through planning, modeling, construction and deployment, culminating in ongoing support for the completed software.

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

- ✓ The process is iterative as the software engineer goes through a repetitive process of requirement analysis-design-testing through prototype implementation –assessment- evaluation, till all users and shake holders are satisfied.
- ✓ In this model once the defect is detected we need to go back to that particular phases where it was introduced& redo some work on that phase& go to the next phase ,so on.
- ✓ For that we need feedback path on the waterfall model& the error should be detected in the same phases.
- ✓ The principle of detecting error as close to its point of introducing as possible is known as “PHASE CONTAINMENT OF ERROR”.
- ✓ This model is simple to understand and use.
- ✓ It is not suitable for very large projects that are subject to many risks.

DIAGRAM:



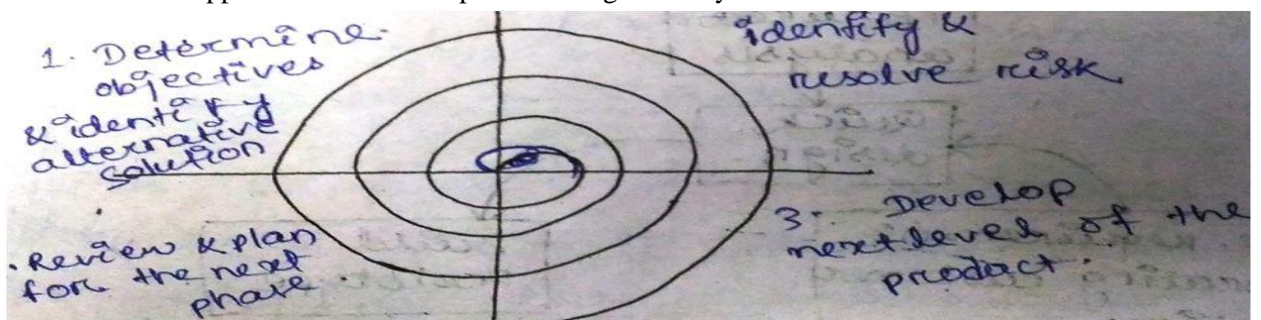
SPIRAL MODEL

This model was developed by Barry Boehm in 1988 There are four phases or four sector in spiral model. Each phase or sector is splitted arbitrarily into four highlighted activities.

- Planning: includes determination of objectives, alternatives and constraints.
- Risk analysis: includes analysis of alternatives, identification and resolve of risks.
- Development: for product development and testing product.
- Assessment: for customer evaluation.

➤ ADVANTAGE:

- It is a cyclic approach for incrementally growing a system's degree of definition and implementation while decrease degree of risk.
- It is a realistic approach to the development of large scale systems and software.

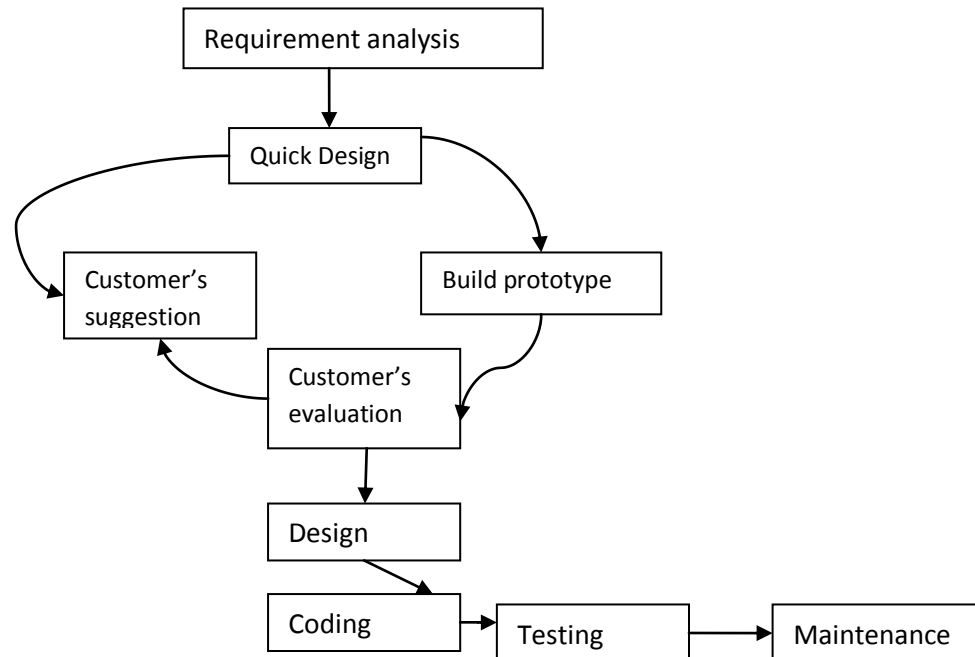


Prototype Model

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

A prototype model is working model of system .it is use to construct this model either when the requirement are not well understood or defined by end user.



RAD MODEL

As the name suggests Rapid Application Development (RAD) model is an **incremental software process model** that focuses on short development cycle time. This model is a “high-speed” model which adapts many steps from waterfall model in which rapid development is achieved by using component based construction approach.

In case if project requirements are well understood and project scope is well known then RAD process enables a development team to create a fully functional system i.e. software product within a very short time period may be in days.

RAD model is like other process models maps into the common and major framework activities.

PHASES OF RAD MODEL

Communication is an activity which works to understand the business problem and the information characteristics that should be accommodate by the software.

In RAD model **Planning** is required because numerous software teams works in parallel on different system functions.

Modelling includes 3 major phases –

1. Business modeling
2. Data modeling
3. Process modeling

Construction focuses mainly on the use of existing **software components** and the application of automatic code generation.

In the last stage **Deployment** establishes a basis for subsequent iterations if necessary.

A business application which can be modularize in a way that allows each major function to be completed in less than three months is useful for RAD. Each major function can be addressed individually by a separate RAD and then integrated to form a whole application.

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

ADVANTAGES OF RAD MODEL

1. Flexible and adaptable to changes.
2. Prototyping applications gives users a tangible description from which to judge whether critical system requirements are being met by the system. Report output can be compared with existing reports. Data entry forms can be reviewed for completeness of all fields, navigation, data access (drop down lists,checkboxes, radio buttons, etc.).
3. RAD generally incorporates short development cycles – users see the RAD product quickly.
4. RAD involves user participation thereby increasing chances of early user community acceptance.
5. RAD realizes an overall reduction in project risk.
6. Pareto's 80 – 20 Rule usually results in reducing the costs to create a custom system.

DISADVANTAGES OF RAD MODEL

1. Unknown cost of product. As mentioned above, this problem can be alleviated by the customer agreeing to a limited amount of rework in the RAD process.
2. It may be difficult for many important users to commit the time required for success of the RAD process.

DRAWBACKS OF RAD MODEL

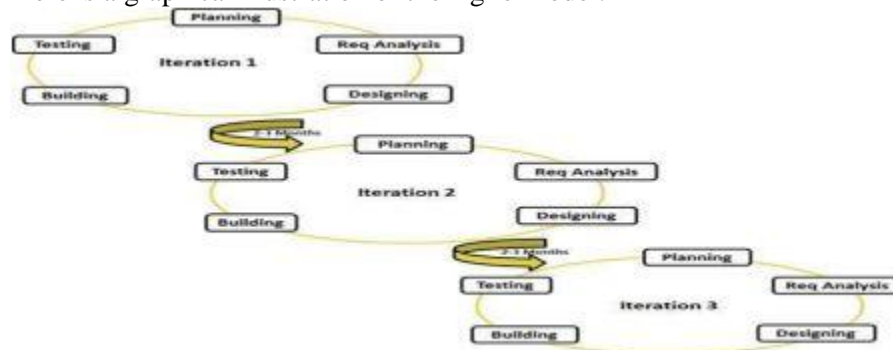
1. RAD requires sufficient human resources to create the right number of RAD teams
2. If developers and customers are not committed to the rapid, rapid-fire activities necessary to complete the system in a much abbreviated time frame, RAD projects will fail.
3. For RAD system should be properly modularize otherwise it creates lots of problems to the RAD model.
4. Rad approach does not work properly if high performance is a major issue, and performance is to be achieved through tuning the interface to system components.
5. When technical risks are high RAD may not be a suitable option. This may be possible while an application heavy uses a new technology.

AGILE MODEL

Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In agile the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.

Here is a graphical illustration of the Agile Model:



Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability.

The most popular agile methods include Rational Unified Process (1994), Scrum (1995), Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, and

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

Dynamic Systems Development Method (DSDM) (1995). These are now collectively referred to as agile methodologies, after the Agile Manifesto was published in 2001.

Following are the Agile Manifesto principles

- **Individuals and interactions** - in agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.
- **Working software** - Demo working software is considered the best means of communication with the customer to understand their requirement, instead of just depending on documentation.
- **Customer collaboration** - As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.
- **Responding to change** - agile development is focused on quick responses to change and continuous development.

Feasibility study

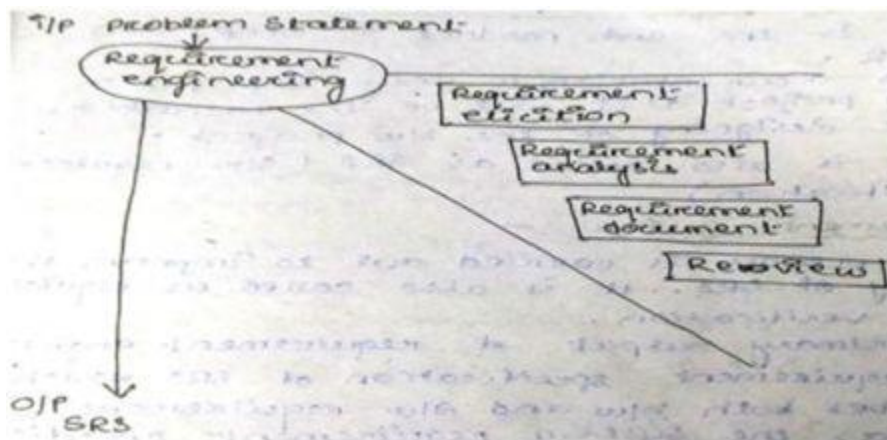
An abstract problem definition is rough description which considers only the important requirement and ignoring the rest, i.e., anomaly, incomplete and consistency.

- Formulation of different solutions out of different solution, most suitable solution will be applied.
- Analysis of alternative solution to examine their benefit and short community.
- Review all the previous process.

REQUIREMENT ENGINEERING

- The objective is to what of the system but not how the system behaves.
- It produce a large document i.e. software requirement specification (SRS) and input of this requirement engineering is the problem definition.

DIFFERENT STEP OF REQUIREMENT ENGINEERING

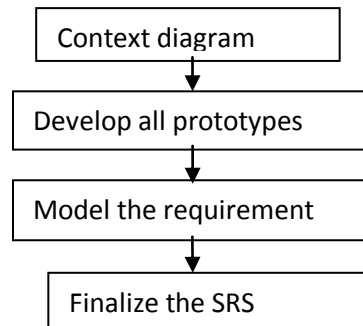


Step 1:- REQUIREMENT ELICIATION

- Also known as information gathering and requirements are identified with the help of customers and existing system.
- This can be achieved by interviewing the customer or questioner, brain storming.

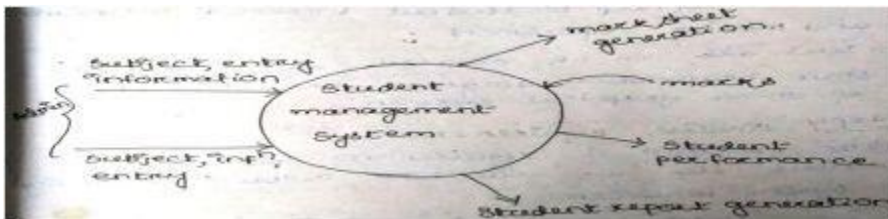
Step 2:-REQUIREMENT ANALYSIS

- It will consist of context diagram.



1. Context diagram is a simple model and define the boundary and interfaces of the proposed s/w system with an external world.

- ✓ It also identifies the entities of outside of the proposed system that interact with the system.



(Fig: Context Diagram for Student Result Management System)

2. Develop the prototype:-

- ✓ This is the effective way to find out what the customer really wants from the system and a prototype is something that looks, preferably acts like a part of the system on which modifications are done by using their feedback until the customers are satisfied.

3. Model the requirement:-

It represent or consist of various graphical representations of functions, data entity, external entities and build a relation between them by the help of DFD(data flow diagram),ERD,DD(data dictionary),structure analysis etc.

4. Finalize the Requirement:-

- It is a large document written in human understandable language.
- It is helpful for future reference if the system is not working properly.

TYPES OF REQUIREMENTS

1. KNOWN REQUIREMENT:-something the end users believe to implement.
2. UNKNOWN REQUIREMENT:-something the end users believe not to implement
3. UNDRAMT:-end user is not able to think about the requirements due to its limitation domain knowledge.
 - 3.1 Functional requirement:-it describes what the system or the s/w has to do and sometimes this requirement is known as product features.
 - 3.2 Non-functional requirement: - these requirements are mostly quality requirement that stipulate that how well the system behaves.

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

Ex: - reliability, flexibility and portability and performance.

CONTENTS OF SRS:-

1. Functional requirement
2. Non –functional requirement
3. Goals of implementation

Traceability: it is process of checking which design component corresponds to which requirement.

CHARACTERISTIC OF GOOD SRS

1. Concise: - irrelevant descriptions should be removed i.e. Ambiguity, incomplete, inconsistency, anomalies.
2. Structured: - well understood format and any time changes can be possible.
3. Black box view/test:-what system does instead of how.
4. Conceptual integration: - how the interface is behaving well.
5. Response to the undesired events:- it also response to the undesired events

CHARACTERISTIC OF BAD SRS

1. Forward reference: - in this we shouldn't refer to aspect that are discussed much later in SRS, which reduced the readability of the specification.
2. Over specification: - over-specification occurs when you try to address the how to aspects in the SRS document. For example-in the library automation problem, you should not specify whether the library membership records need to be stored on the member's first name in descending order arrangement. Over-specification restricts the freedom of the designer in arriving at a good design solution.
3. Wishful thinking: - this type of problems concern description of aspects which would be difficult to implement.
4. Noise: - the term noise refers to presence of material not directly relevant to the software development.

Organization of SRS

1. Introduction
 - a. Background
 - b. Overall description
 - c. Environmental characteristics
 - i. Hardware
 - ii. Peripherals
 - iii. People
2. Goal of implementation
3. Functional requirements
4. Non-functional requirements
5. Behavioral description
 - a. System state
 - b. Event and action
6. Conclusion
7. Reference/Bibliography

Different users of SRS

❖ **End users**

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

The goal of these users to ensures that the system as describes in SRS will meet their needs.

❖ **Software developers**

These persons refers SRS to make sure that they have developed exactly same as described in the SRS.

❖ **Test engineers**

The goal is to understand from the functionality part of view so that they can test the software and validate.

❖ **User document writer**

The users are reading the SRS and ensure that they understand the document and then only they are able to create the user manual.

❖ **Project manager**

They can estimate the cost of the project by referring the SRS.

❖ **Maintenance**

These are the users by referring the SRS they can maintain the software in future if needed.

Structure analysis/design methodology

- ❖ It is sometimes called as structure oriented software design.
- ❖ It is divided into 2 parts.

1. **Structure analysis**

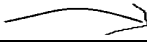

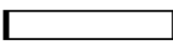
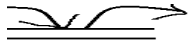
- a. Goal of structured analysis is to analyze the functionality of system and data flow among proceeding task which are represented graphically.
- b. The software analysis based on following principles:
 - i. Top-down approach
 - ii. Bottom-up approach
 - iii. Graphical representation in the form of DFD(Data flow Diagram)

2. **Structured design**

- ❖ The aim of the structure design is to convert the result of structure analysis in the form of structure chat which represents the software architecture in the form of module making of the system module dependency.

DFD (Data Flow Diagram)

1. It is a graphical representation of flow of data through information which is widely used for modeling the requirements in SRS.
2. It shows the flow of data to a system where the system may be a company or an organization. It also known as bubble chart.

| Symbol | Name | Description |
|-------------------------------------------------------------------------------------|--------------|------------------------------------------------------------------------|
|  | Data flow | Used to connect source to destination. |
|  | Process | Perform some transformation of input to get some output. |
|  | Source /sink | A source is the system input and sink represent output. |
|  | Data store | Repository of data. The arrow head indicates the net input and output. |

Characteristics of DFD

1. All names should be unique.

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

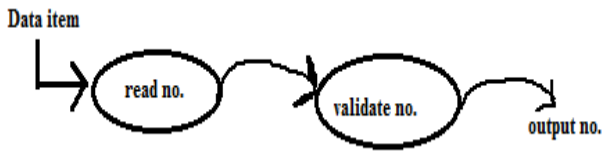
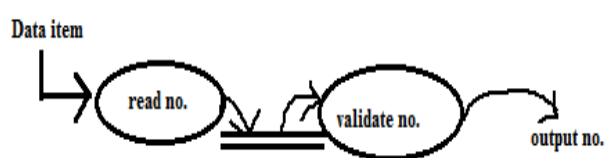
PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

2. DFD is not a flow chat, where the arrow in the flow chat represents the order of the events whereas in DFD it represents flow of data.
3. In DFD we can't take decision.
4. In DFD data can be stored using data-store.

Types of DFD

There are 2 types of DFD

- ❖ Synchronous
- ❖ Asynchronous

| Synchronous | Asynchronous |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| If two bubbles are directly connected by using dataflow then it is called as synchronous. | If two bubbles are directly connected by using data store then it is called as asynchronous. |
|  <p>The diagram shows two data stores, 'read no.' and 'validate no.', connected by a data flow arrow. An external data item 'Data item' enters the 'read no.' store, and an 'output no.' exits from the 'validate no.' store.</p> |  <p>The diagram shows two data stores, 'read no.' and 'validate no.', connected by a data store symbol (two parallel lines). An external data item 'Data item' enters the 'read no.' store, and an 'output no.' exits from the 'validate no.' store.</p> |

Leveling of DFD:

1. Level 0 or L0 or context diagram/fundamental system model. This shows the interaction between the system and external agent.



Level1:L0 is now being explored into different section and so on.

Example: DFD for Calculating RMS.



Data Dictionary

Data dictionaries are simply repositories to store information about all data items defined in DFDs. Different terminology is used in data dictionary are follows.

1. Name of Data item.
2. other name for item(Aliases)

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

3. Description/Purpose.
4. Range of values.
5. data structure information(DS)

Mathematical representation of data dictionary.

Notation

1. $x=a+b$
2. $x=[a/b]$
3. $x=a$
4. $x=y\{a\}$
5. $x=\{a\}z$
6. $x=y\{a\}z$

Meaning

- 'x' consists of data item 'a' & 'b'
- 'x' consists of data either 'a' & 'b'
- 'x' consists of optimal data item 'a'
- 'x' consists of y or more occurrence of 'a'
- 'x' is consists of z or fewer occurrence of 'a'
- 'x' is some occurrence of a between y & z

Use of DD

- To create an order listing of all data items.
- Listing of a subset of data item.

Ex: DD for RMS Calculating S/W

Data Items :{ Integer } 3

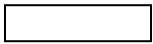


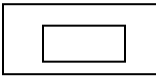
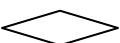

RMS: Float

Validate data: Data item

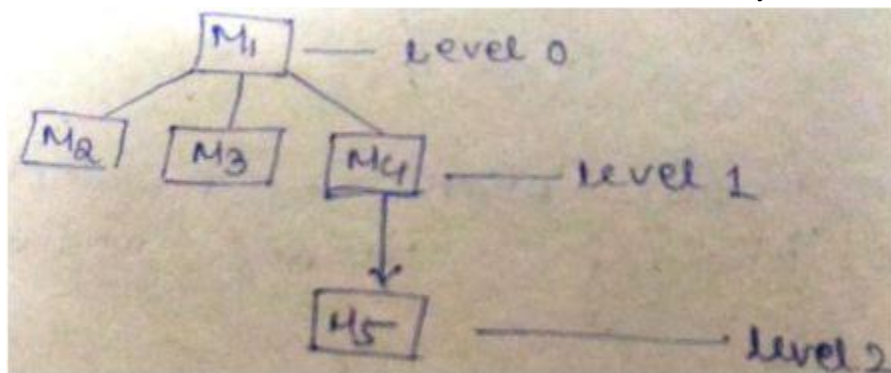
a : integer, b : integer, c : integer , a sq: integer , b sq: integer , c sq: integer ,m sq: integer

SA/SD: The result of SA/DFD will converted into structure chart in SD.

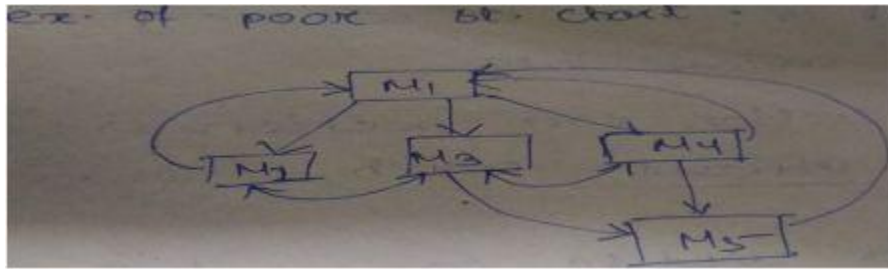
Some symbols are used for representation of SA are follows:

| | |
|-------------------------------------------------------------------------------------|------------------------------------------------------------------|
|  | This represents a module which are usually represents |
|  | Module invocation |
|  | Small arrow appearing alongside the module is known as data flow |
|  | Library module |
|  | Selection |
|  | Root/parent module |

There should be at most one control relation between any two modules



Layer structure of module



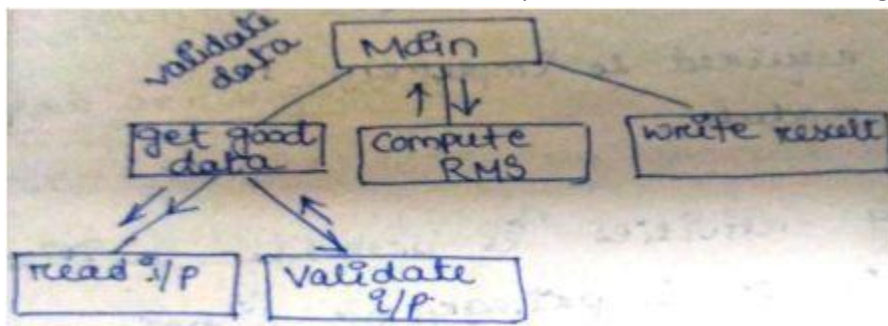
DFD to Structure Chart:

There are two ways to convert DFD to Structure Chart.

1. Transform Analysis (TA)
2. Transaction Analysis

1. Transform Analysis (TA):

- It identifies the primary functional components or module & the high level i/p or o/p.
- first step of transform analysis is divided in three parts
 - Input: used to convert physical data into logical data. in input section each portion is called as **afferent branch**.
 - Logical processing: it is used to do all logical operation in between i/p to o/p.
 - Output: used to convert logical to physical data and each portion of output is called **efferent branch**.
- Step 2 of TA is that the structure chart is divided by drawing on functional component for control transform and the afferent & efferent branches.
- In the 3rd step of TA, the structure chart is defined by adding sub function or module required by each of the higher level functional component.
- Some time these sub-functional components are called as **factoring**.



(Fig: Structure chart for RMS Calculating S/w)

Transaction Analysis: it is an alternative solution to TA and is useful while designing the transaction processing program.

- In transaction analysis system are of several possible path through the DFD is traversing depending upon the i/p data.

S/W Design:

During the designing phase of soft ware (S/W) the following items must be design

1. Different module requires implementing the designing solution.
2. Control relationship among the module.
3. Interface among the module.

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

4. Data structure of individual module.
5. Algorithm required to implement.

Characteristic of good S/W engineering;

1. Correctness. 2. Understandability. 3. Efficiency. 4. Maintainability.

Terminology used in s/w design:

1. Clean decomposition: means module in s/w design should be follow mechanism of high cohesion and low coupling.
2. Neat arrangement.
 - (a) Layered solution: modules are arranged in different layers.(Example layered is in the previous page)
 - (b) Low fan out and high fan in: low fan out means the measure of number of module that is directly controlled by a given module. A design having high fan out not required. Whereas fan in indicates the number of modules directly calling a module so high fan in represents code reuse or reusability.
 - (c) Abstraction.

Cohesion and Coupling

- Cohesion is a measure of functional strength of module.
- Coupling is a measure of functional strength between modules.

A module having high cohesion and low coupling is said to be a functional independent of other module.

Types of cohesion

1. Coincidental (ex; random collection of function)
2. Logical(Ex: Error handling)
3. Temporal(Ex: initialization and shut down of system)
4. Procedural(ex: algorithm for encoding and decoding of message)
5. Communicational(Ex: function s defined on stack)
6. Sequential (Ex: module 2 taking the output of module 1)
7. Functional (Ex: functions that manage payroll of employee)

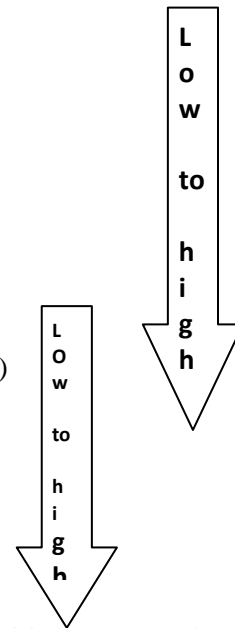
Types of coupling;

1. Data coupling(Ex: int /char passed as parameter from one module to other)
2. Stamp coupling(Ex: structure ,union)
3. Control coupling(Ex: flag set)
4. Common coupling (Ex: global variable)
5. Content coupling (same code).

Design Approach

There are two types of design:

1. Function oriented: A system is viewed as something that performed a set of functions starting at high level view of system and each function again subdivided called as sub function. Here system state is centralized and shared among different function.
Ex: library function is further divided as **a**) assign membership function **b**) display all record, etc.
2. Object oriented: system is viewed as a collection of object and system state is decentralize.



Module-II

OBJECT ORIENTED CONCEPT AND PRINCIPLE

Key concept of Object oriented design:

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

- a) Abstraction (b) Polymorphism (c) Encapsulation (d) Dynamic binding (e) Genericity
(f) Composite object

Concept (a,b,c,d) is same as in C++ /JAVA and OOP.

e) **Genericity** : is the ability to parameterize class definition i.e. while defining class stack of different element such as integer stack, character stack etc genericity permits as to define a generic class of type stack etc.

f) Object which contains another object is known as composite object which can be used for complex behavior of object.

Related terms:

- **Agents**: Is the active object, which monitors events occurring in the application and take the action autonomously. ex : database application
- **Widget**: stands for window object and is a primitive object which is used in the design of GUI.
Ex: Background, foreground etc.

PRESISTANCES

Objects are usually get destroyed once a program finishes its execution where as persistence object are stored permanently in the secondary storage device.

Eg:- Is a relationship

Q. What is “is a relationship”?

Ans: “is a relationship” corresponds in oops concept is inheritance.

For e.g.: Classroom room is a room

Class classroom extends room

// Body of Class

Q what is “has a relationship”?

Ans: “has a relationship” in oops concept corresponds to encapsulation.

For ex: classroom is a room which has a blackboard.

Class Classroom {

bb b1 = New bb ();

Class bb

}

Object Oriented Modeling Techniques:-

- (1) Booch’s object oriented modeling Technique.
- (2) Rumbaugh’s object oriented modeling Technique.
- (3) Jacobson’s design model.

All 3 (Three) techniques are now converted to UML diagram. (Unified Modeling Language)

Booch’s Object Oriented Modeling Technique:-

Booch’s object oriented modeling techniques is also known as object oriented analysis and designing (OOAD) and it is a precursor to the UML program and which includes six type of diagram i.e. Class diagram, Object diagram, State transition diagram, Interaction diagram, Module diagram & Process diagram

Booch’s class & Object diagram notation

- (1) Class
Booch use cloud shape with dash border.

(2) Object

Booch use cloud shape with solid border.



(3) Class adornments :

Use adornment to provide additional information about a class to create an adornment by using some letters inside the triangle that will be an additional information about the class.

F-> Friend

i.e. a friend class allow to access any non public member.

A->abstract , S->static , V->virtual class

Meta class

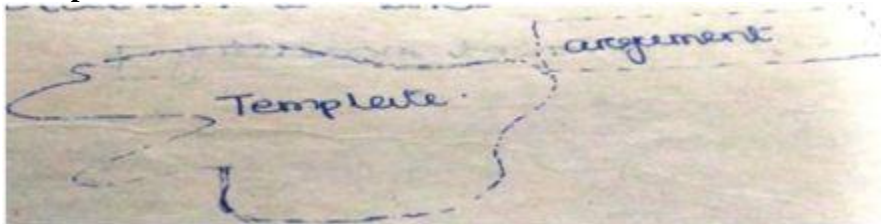
It is class whose instances are also a class. The notation used is same as class notation.

Class categories

Represent a cluster of similar class. The notation is a rectangular with two components.

| |
|-----------------------|
| Class Categories name |
| Classes |

Class template



Class Utility

Describing the groups of non-member function or sub programmes with shadowed cloud.



Class Visibility

The visibility will place next to the class attribute.

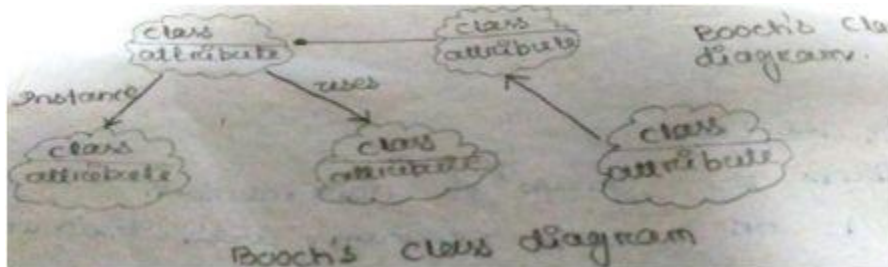
- (1) Private (2) Protected (3) Public

Object Visibility

The visibility maker will be placed on a link, to signing the relationship between the connected object and the makers are

- (1) G – Global (2) P – Parameterized (3) L – Local

Ex: - (1) [Booch's object oriented modeling]



The relationship symbols used in booch diagram are:-

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

- (i) Aggregation [has a membership] (ii) Aggregation by value (iii) Aggregation by reference
- (iv) Uses [use a relationship] (v) Instantiate (vi) Instantiate new type (vii) Inheritance
- (viii) Inheritance with new type

Booch's dynamic diagram:-

Uses state transition diagram and interaction diagram for dynamic nature of an application where state transition diagram is equivalent to state chart in UML diagram and interaction is same as sequence diagram in UML diagram.

Ex:-

| |
|------------|
| Name |
| Action (H) |

Here Name indicates state and H indicates adornment.

Rumbaugh's object oriented modeling technique:-

It is one of the pre-cursors to the UML diagram. There are 3 (Three) diagram i.e.

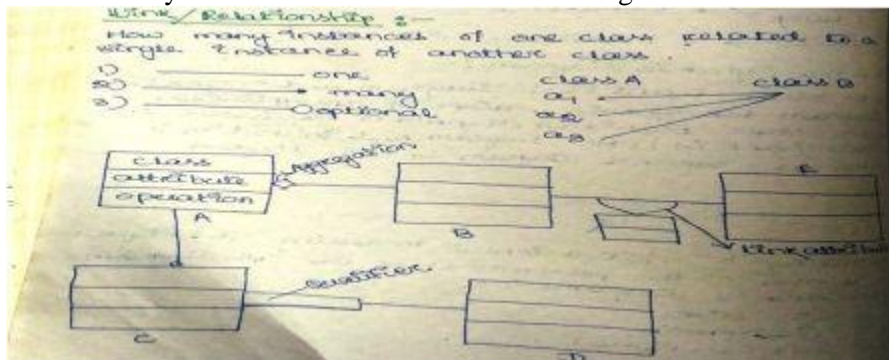
- (i) Object (ii) Dynamic (iii) Functional

Object Diagram

It illustrates the static relationship among classes and object in a system. These diagram is similar to the object and class diagram of UML.

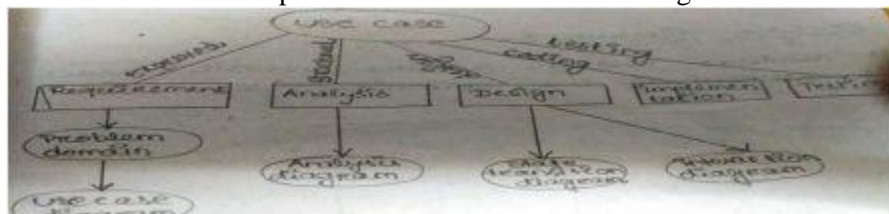
Link/ Relationship:-

How many instances of one class related to a single instance of another class?



Jacobson's Model:-

Jacobson's model includes requirement analysis the designing & implementation & testing phases/ model & the relationship between the central use case diagram & various models is as the diagram below.

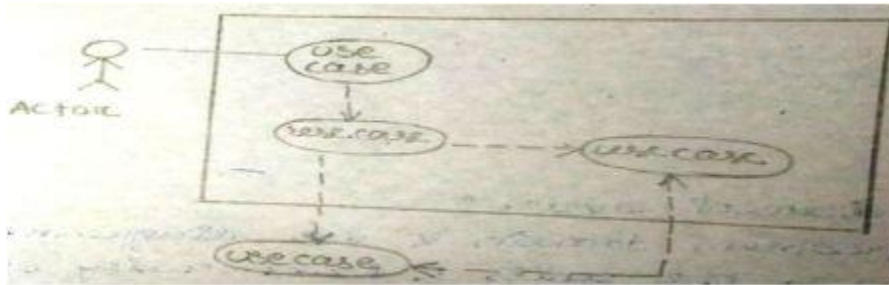


Jacobson's Requirement Models:-

It includes problem domain & use diagram. This model defines the units & functionality of a system. Whereas the problem domain object diagram provides a logical view of a system which is used to specify the use case on use case diagram.

Use-case Diagram:-

It describes how the outsiders, World interact with element of the application system.



(fig: schematic diagram of use case)

UML (Unified Modeling Language)

- UML provides set of notations to create models of system where the models are very useful in documenting the design and analysis result. Also a model facilitate the generation of analysis and designing procedures.
- UML develop to standardize large no. of object oriented modeling notation & was used extensively in 1990.
- Schematically representation of different object modeling



(fig: Schematic representation of different object model)

UML was developed by OMG (object Management Group)

What is Model:-

A model captures aspects important for some application while omitting the rest or a model in the context of S/W development can be represented in the form of graphical, mathematical or a code based textual.

Why to construct a model?

An important reason behind constructing a model is that it helps to manage the complexity of S/W/ development.

Once the model of a system have been constructed than these can be used for a variety of purpose, which includes

- (i) Requirement analysis (ii) Specification (iii) Code generation (iv) Design
- (v) Visualize (vi) Testing

UML Diagram

It can be used to construct 9 (nine) different diagrams i.e.

- (1) Use-case diagram (2) Class diagram (3) Object diagram (4) Sequence diagram
- (5) Collaboration diagram (6) State chart (7) Activity diagram (8) Component diagram
- (9) Deployment

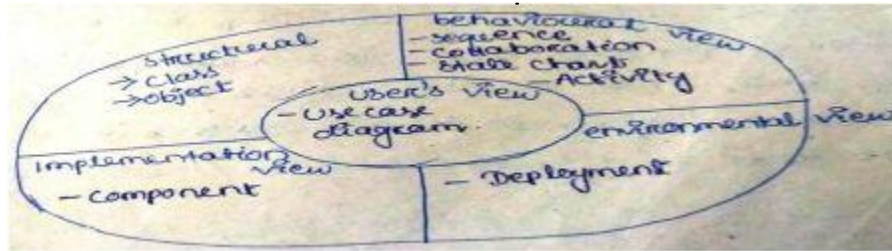
The different UML diagram provides different activity of S/W system to be developed & facilitate a comprehensive understanding to the system such models can be fine to get the actual implementation of the system.

The UML diagram can capture the following views of a system.

- (1) User's view (2) Structural view (3) Behavioral view (4) Implementation view
- (2) Environmental view

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE



(fig: UML Diagram)

1) USER'S View:-

This view defines the functionalities made available by the system to its user.

- It captures the external users views of the system in terms of function offered by system. This view is a black box view. Because it will not showing the internal functionalities because in this view the internal features are not visible.
- This view is considered as central view and all other views are expected to confirm to this view.

2) Structural view

It destines the kind of objects of class imp to the understanding of the working of a system to its implementation.

- It also captures the relationship among them .Sometimes this view is also called as static view or model since the structure of a system doesn't change with the time.

3) Behavioral view

This view catches how objects interval with each other to realize the system behaviors

- This view is also called as dynamic model since the behaviors of the system changes with time hence it is called as dynamic model.

4) Implementation view

This view captures the important components of the system & their interdependencies.

5) Environmental view

This view models how the different components are implemented of different piece of hardware.

Use case model

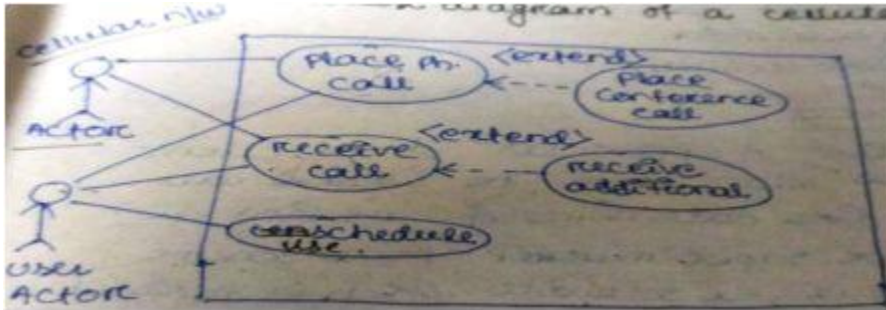
- The use case model or diagram for any system consists of a set of use case, i.e. a use case represents the diff. ways in which a system can be used by the users.
- A simple way to find out ale the use cases of a system is to ask the question like what the uses can do using the system.
- E.g. Library management system.
 - i) Issue books (ii) Quickly book (iii) Create a membership (iv) Add book.
- The purpose of a use case is to define a piece of coherent behavior without revealing the internal structure.

Representation of use case:-

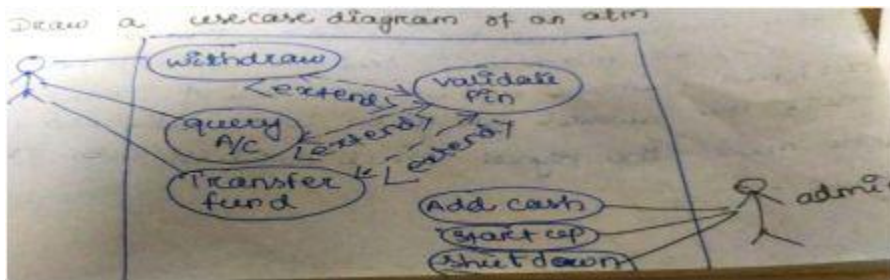
A use case can be represented by drawing a use case diagram and working a text elaborating a drawing

- In the use case diagram each use case is represented by an ellipse and an ellipse with the name of the use case written inside the ellipse and all ellipses are enclosed within a rectangle which represents a system boundary.
- Diff. Uses of the system are represented by using a stick person. Icon which is referred as an actor where an actor is a role played by a user with respect to the system use.

DRAW A USE CASE DIAGRAM OF A CELLULAR N/W



DRAW A USE CASE DIAGRAM OF AN ATM



Text description:

Each ellipse on the use case should be accompanied by a text description. It should define the details of interacting between the users and the compeller. It should include all the behavior associated with the use case. The following are some type of information which may be includes in use case text description in addition to maintain sequence and scenarios.

1. Actor
2. Contact person
3. Pre condition
4. Post condition

Contact person

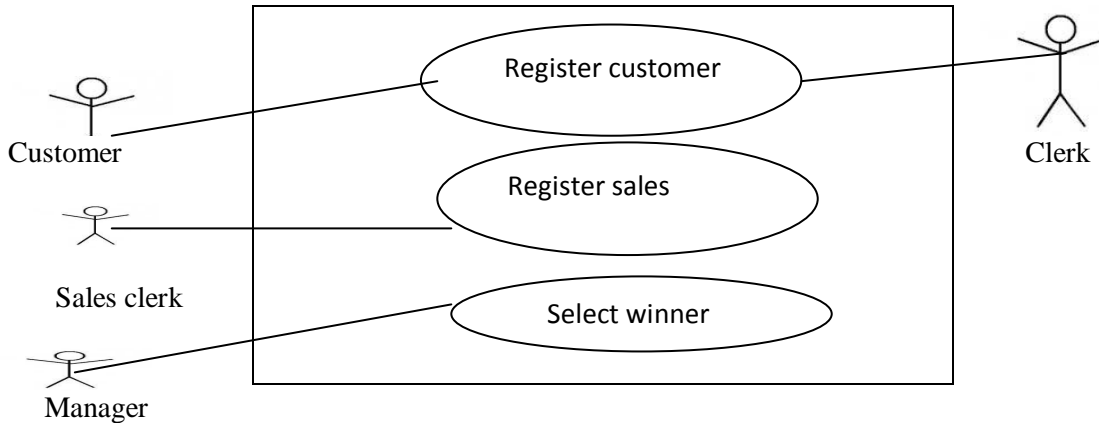
In this list of personals of the client organization with whom the use case was discussed like date, time of muting etc.

Ex: - For super market prize scheme. In this scheme, software which is assigned by a customer number to the computers and every purchase made by the user, the value of his / her purchase is credited against his cues tamer number and at end of each year; the award was given to 10 customers who makes the highest total purchase over the year.

Q. Design a use case diagram for this particular supermarket prize scheme according to the above description.

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE



Text description:-

Using this use case, the customer can register himself by providing necessary information of details (user1).

Scenario 1:- In this case, select register customer option.

2. System: - Display the prompt to enter the address, name, contact details.

3. Customer: - Enter the necessary values.

4. System: - Display the generated id and the message that the customer has been successfully registered.

Scenario2:-

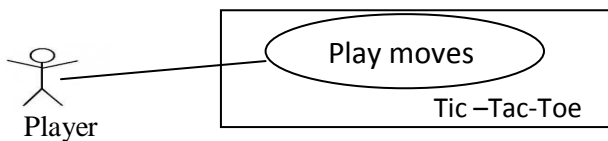
At the step4 of scenario 1, the system displays the message that the customer has already registered.

Scenario 3: -

At the step4 of Scenario1, the system displays the message that same input information has not been entered.

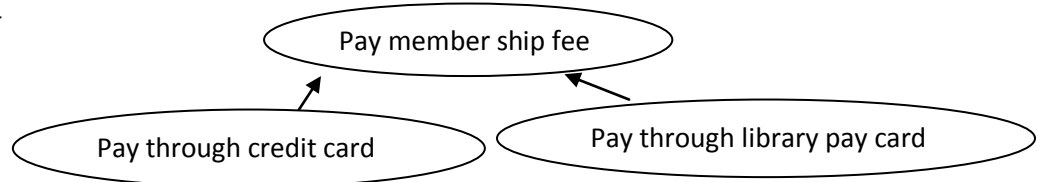
How to identify use case

Identification of use case involves brainstorming & viewing the existing SRS, where as an actor base method of identifying the use case is quite Popular, In this case we have to 1st identify the diff. actor.



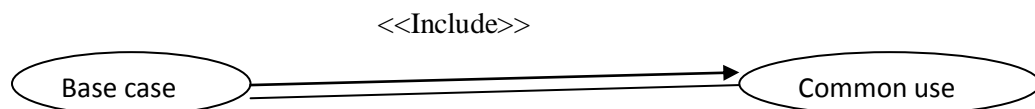
Factoring of commonality among the use case:-

1) Generalization:-It can be used when you have one use case similar to another but does something slightly different-



2) **Includes**

It is known as using a reshpi. It involves one use case including the behavior of another use case in its sequence of even & actions.



3) **Extend**

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

The main idea behind the extended relationship among the use cases is that it allows showing the optional system behavior.

- Where as an optional system behavior is executed only under certain Condition.

<extend> Use case diagram



Class diagram

- A class diagram describes the starting structure of a system and it shows how a system is structured rather than how it behaves whereas the static structure of a system consists of a no. of classes and their dependency.
- The main constituent of a class diagram is classes and their relationship.
- Where the symbol of a class is a rectangle having three components.

| |
|---------------|
| Name of Class |
| Attribute |
| operation |

- Whereas the class name is written in the center of the rectangle with bold letter & also is a combination of upper cases & lower cases or mixed case.

Basically the class names are chosen from SRS in singular nouns.

E.g.: In LIS (Library Information System)

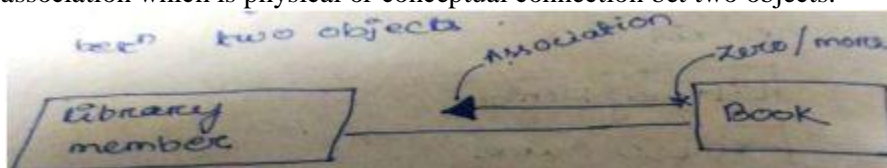
| |
|-----------------------------------------------------------|
| Library member |
| Member name ph.no address |
| Issue book () ; return_book () ; query_book () ; |

1) Generalization- (see use case diagram)

2) Association:-

Associations are needed to enable objects to communicate with each other.

- An association describes connection between classes.
- The relation between two objects is called object connection or link whereas links are instances of association which is physical or conceptual connection between two objects.



GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

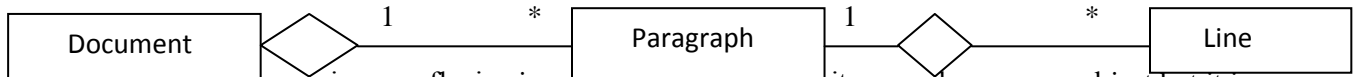
PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

The relationship always identify from SRS by looking the verbs of the SRS.

3) Aggregation:-

It is a special type of association where the involved classes represents a whole part relationship.

- The aggregate takes the responsibility of towards the messages to the appropriate part.
- If an instance of one object contains instance of some other object then aggregation relationship will exist-
- Aggregation relationship is represented by a diamond. Symbol.



Aggregation can't be recursive or reflexive i.e an object can't contain its own class or own object but it is a transitive relation.

4) Composition :-

It is same as your aggregation in which the parts are existence dependence

- i.e. the life of each part is closely tied to the part of whole.
- When the whole is created automatically apart is created
- When whole is destroyed
- The symbol of composition is a solid diamond.

Dependency

It is a form of association between two classes.

- A dependency relation shows a change in the dependent classes requires a change to be made in the independency class.

Object diagram

Shows the snapshot of the objects in a system at a point in time. Since it shows instances of classes, rather than the class themselves. It is often called as instance diagram.

- An object diagram may undergo continuous change as the execution produces. The object are drawn by rounded rectangle.

| |
|----------------|
| Library member |
| Attribute |
| Operation. |

Interaction diagram (ID)

Introduction diagrams are the model that describes how the group of objects collaborates to release some behaviors of the system.

- Physically each interaction diagram realizes the behavior of a single use case and this ID is divided into 2 parts.
 1. Sequence Diagram
 2. Collaboration

Sequence Diagram:-

The sequence diagram shows the interaction among the object as a 2D chart.

- The chart is read from top to bottom.
- The objects participating in the interaction all shown at the top of the chart as box attached to a vertical dashed line
- Inside the box the name of the object written with a colon separating it from the name of the class.
- Both the name of the class & object are underlined.



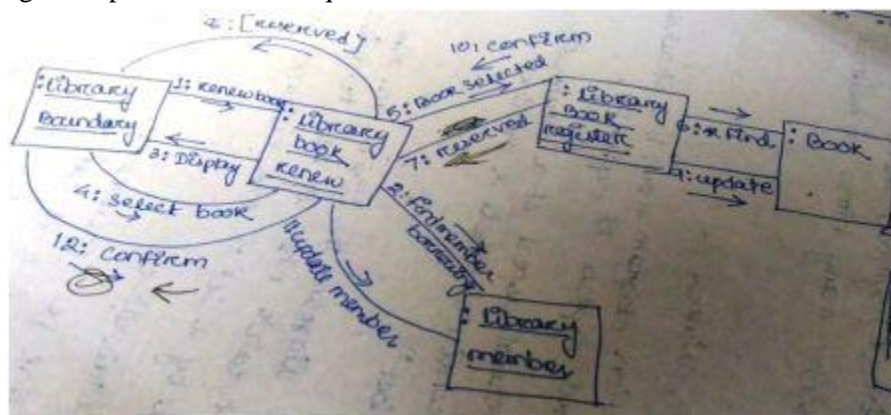
(fig: Interaction Diagram for book renewal in Library Information System (LIS))

- The vertical dash line signifies the object lifetime.
- The rectangle signify activation of object
- Each message is leveled with message name.
- Some control information can also be included with that message.
- There are two types of control information
 - 1) A condition should be indicated that a message is sent only if the condition is true. That condition should be indicated in the square bracket.
 - 2) An iteration (R) symbol shows that the message is sent many times to multiple objects.

Collaboration diagram:-

A collaboration diagram shows both structural & behavioral aspect explicitly.

- The structural aspect of a collaboration diagram consists of objects & the links existing between.
- In this diagram the objects are also called as collaboration where as the behavior described by a set of message exchanged among different collaborators.
- The link between collaborators is shown as a solid line & it can be used to send the message between two objects.
- Whereas the message leveled along with a link.
- And the messages are prefixed with a sequence no.



(fig: collaboration Diagram of book renewal in LIS)

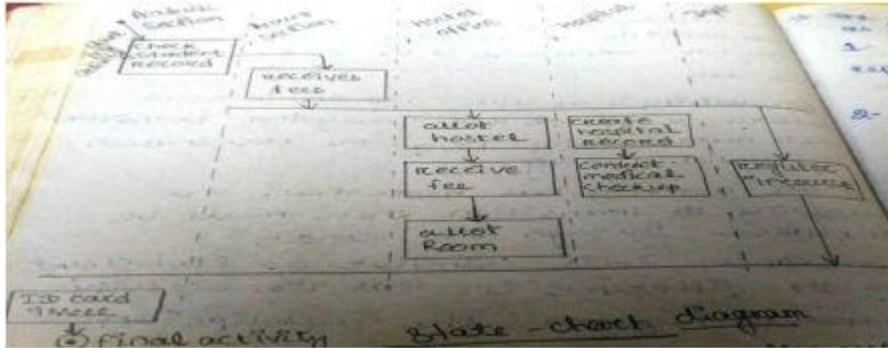
Activity Diagram:-

The activity diagram focuses on representing the activities of processing which may or may not correspond to the methods of the class.

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

- ➔ An activity is a state with an internal actions & one or more outgoing transitions which automatically follow the termination of internal activity .
- ➔ If an activity has more the one outgoing transition these must be identified through the condition.
- ➔ Sometimes it is similar as flowchart but the difference is that the activity diagram supports the description of parallel activities.
- ➔ An important feature of A D is
Swim lanes: - It enables you to group of activities based on who is performing them (objects).
- ➔ This diagram normally employed in Business process modeling.

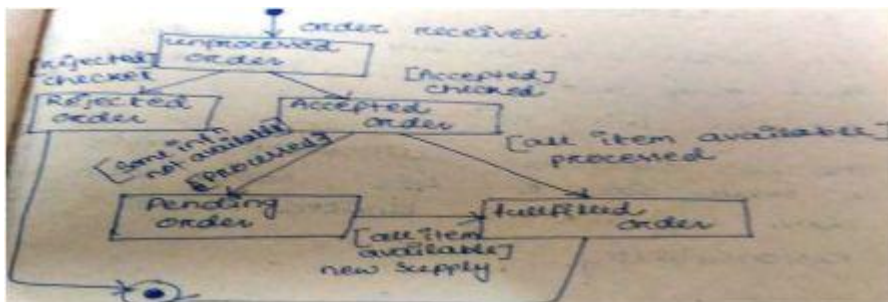


(fig: Activity Diagram for hostel admission process)

State-Chart diagram

- ➔ A state chart diagram is normally used to model how the state of an object changes in its lifetime.
- ➔ These diagrams are good at describing how the behavior of an object changes across several use case execution.
- ➔ This diagram is based on FSM (final state machine)
- ➔ The basic element of a state chart are as follows

| | | |
|---|----------------------|--------------------|
| 1 | Initial state | ● Filled circle |
| 2 | Final state | ○ (●) |
| 3 | State | □ |
| 4 | Transition | → |



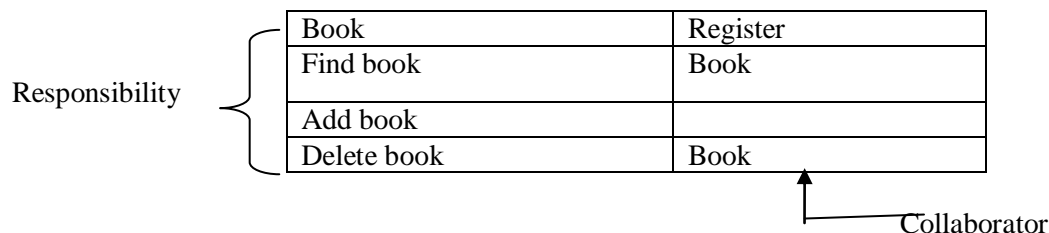
(Fig: State Chart Diagram for online purchasing an Item)

CRC (Class responsibility collaborator)

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

- ➔ The CRC technology was developed by ward Cunningham & kent Beck at the research laboratory of Tektronix at Portland, Oregon, USA .
- ➔ A CRC is also sometimes called as CRC card which are index card that are prepared one per each class.
- ➔ On each of these cards, responsibility of each class is written briefly the object with which another object needs to collaborate its responsibility.
- ➔ CRC cards are usually developed in small group session where people role play being various class.
- ➔ The CRC cards are deliberately made a small size of 4*6, So that each class can have only limited no. of responsibility.



Advantages of object oriented design (OOD)

- ➔ Code & designing reuse
- ➔ Increased productivity. - It is possible due to the code reuse by using some predefined class & libraries.
 - It is also possible due to the inheritance & better problem decomposition.
- ➔ Easy of testing & maintenance.
- ➔ Better code & designing understandability.

OOD goodness Criteria:

- ➔ Coupling guidelines (ii) Cohesion guidelines

MODULE -III CODING & TESTING

Coding & Testing: coding is under taken once the design phase is complete and design documents have been successfully reviewed.

- The input to the coding phase is the design document.
- During this phase different modules identified in the design document are coded according to the respective module specification.
- Good software development organizations require their programmers to adhere some well defined and standard style of coding called coding standard & their engineers to follow these standards rigorously due to the following reasons.
 - A coding standard gives a uniform appearance to the codes written by different engineers.
 - It provides sound understanding of the code.
 - It encourages good programming practices.
- Besides these standards several coding guidelines are also suggested by software companies.
Coding Standards and guidelines
Some of coding standards and guidelines which are commonly adopted by many software companies:
 1. Representative coding standards: the following are some representative coding standards.

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

- a. Rules for limiting the use of globals: These rules list what types of data can be declared as global and what cannot.
- b. Contents of headers preceding codes for different modules :the information contained in the headers of different modules should be standard for an organization. following some standard header data:
 - i. Name of the module.
 - ii. Data on which the module was created.
 - iii. author's name
 - iv. Modification history.
 - v. synopsis of the module
 - vi. different function supported
 - vii. Global variables accessed.
- c. naming convention for global variables and constant modifiers:
- d. error return conventions and exception handling:

Representative coding guidelines:

1. do not use a coding style that is too clever or too difficult to understand
2. avoid obscure side effects.
3. Do not use an identifier for multiple purposes.
4. Code should be well documented.
5. the length of any function should not exceed 10 source lines.
6. do not use goto statements.

Code review:

This was carried out after the successfully compiled and the syntax error eliminated and is cost effective strategies. There are two types of code review

1. code walk-through
2. Code inspection.

Code Walk-Thorough:

- It is an informal code analysis technique. In this technique the module has been coded and is successfully compiled and all syntax errors are eliminated.
- The main objective of it is to discover the algorithm and logical errors in the code.
- some of the guidelines are:
 - The team performing the code walk- through should not be either too big or too small. Ideally, it consists of three to seven members.
 - Discussion should focus on discovery of errors and not on how to fix the errors.
 - In order to foster cooperation and to avoid the feeling among the engineers that they are being evaluated in the code walk-through meeting, managers should not attend the meeting.

Code Inspection:

- the aim is to discover some common type errors caused due to oversight and improper programming
- list of some classical programming errors which can be checked during inspection:
 - use of uninitialized variables
 - jumps into loops
 - non terminating loops
 - incompatibility assignments
 - array indices out of bound
 - improper storage allocation & deallocation
 - mismatch between actual and formal parameters

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

Clean room testing:

It is pioneered by IBM and this type testing relies heavily on walk-throughs, inspection and formal verification. Here the programmers are not allowed to test any of their code by executing the code other than doing syntax testing using compiler.

Soft ware Testing:-

- The objective of software testing is to identify all the defects existing in the software product.
- Testing a program consists of subjecting the program to asset of test i/ps or test cases & observing, if the program behaves as expected and if fails to behave as expected then debugging and correction is required.

COMMON TERMS

Failure: a failure is a manifestation of an error or defect/bug, but the mere presence of of an error /a bug/a defect may not necessarily lead to a failure.

Test case: A test case is the triplet of [I , S, O], where I: is data i/p to the system.

O: is o/p to the system.

S: is the state of system.

Test Suite: it is the set of all test cases with which a given S/W product is to be tested.

Verification: it is the process of determining whether the o/p of one phase of s/w development confirm to that of its previous phase.

- It is concerns with phase-contaminated error.

Validation: it is the process of determining whether it is fully developed system or product confirm to its requirement specification.

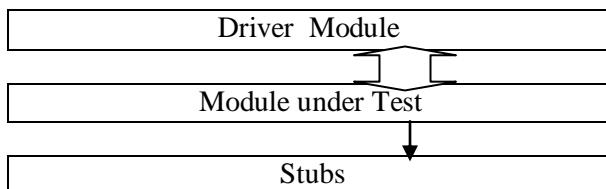
- The aim of this process is to find an error free final product.

Testing in large Vs Testing in small

| Small | Large |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none">• S/W product is normally tested first at their individual components/unit level.• This is sometimes called unit testing. | <ul style="list-style-type: none">• After testing all components are slowly integrated and tested at each level of integration.• Called as integration testing.• After integration testing the final product goes for system testing. |

Unit Testing: In order to test a single module we need a single & complete environment to provide all that is necessarily for execution i.e beside the module under test, we will need the following order to be able to test the module.

- The procedures belonging to other modules that the modules under test cases.
- Nonlocal data structure that the module access.
- The procedure to call the functions for driver module and stubs.



Driver Module

This module contains the nonlocal or global Data structure & also has code to call different function of the module.

Stub Module

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

Stub procedure is a dummy procedure that has same i/o parameter as the give procedure but has highly simplified behaviors.

Black Box Testing:

- Test cases are designed from an examination of i/p or o/p value only and no knowledge of designing or coding is required.
- Basically the test cases are designed using only the functional specification of software product i.e. without any knowledge of internal structure of the software.
- Hence it is also called as function oriented testing.

Approaches /Strategies:

1. Equivalence class partitioning
2. Boundary value analysis.

Equivalence class partitioning:

- In this approach the domain of i/p value to a program is partitioned into a set of equivalence classes and this partitioning is done in such way that the behavior of a program is similar to every i/p data belonging to the same equivalence classes.
- If the i/p values to the system can be specified by a range of values then one valid & two invalid equivalence classes should be define.
- if the i/p data assumes value from a set of discrete members of some domain then equivalence class for a valid i/p & another invalid i/p should be define.
- Ex: the program compute the intersection point of two lines & display the result on which it reads the value m_1, c_1 & m_2, c_2 for the line equation $y=mx+c$.

The following equivalence classes are

1. parallel line ($m_1=m_2, c_1 \neq c_2$)
2. Intersection Line ($m_1 \neq m_2$)
3. coincident Line ($m_1=m_2$ & $c_1=c_2$)

Selecting one representative value from each class we will get the test cases /test suite $\{(2, 2), (2, 5)\}$ here $m_1=m_2$ and $c_1 \neq c_2$ so parallel Lines.

Boundary Value:-

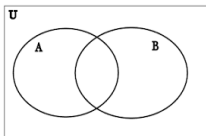
A type of programming error frequently occurs at the boundary of different equivalent classes of i/ps.This is due to the psychological factor of a programmer.

→ The programmers often help to see the special processing required by input values that lye at the boundary of diff. classes.

e.g. Programmer may improperly use ' $<$ ' symbol instead of ' $<=$ ' symbol.

Summary of Black Box Testing:

- Examine the i/p and o/p
- Identify the equiv. classes
- Select the test cases to corresponding equiv. classes and testing with the boundary values.

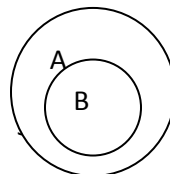


White Box Testing:

It requires a thorough knowledge of internal structure of s/w; hence it is otherwise called as structural testing / glass testing.

- One White Box Testing is said to be stronger than other White Box Testing if all types of errors detected by the first testing are also detected by the second testing.

A is stronger than B



Complementary

1. Statement Coverage:-

Its aim is to design test case that every statement in a program is executed at least once.

- The main idea behind this strategy is that unless we execute a statement we have no way to determine if an error exists in that line.
- However executing a statement once and observing that it behaves properly for that i/p value is sure that it will behave correctly.

```
int GCD(int x,int y)
{
    1. while(x!=y) {
    2. if(x>y){
    3. x=x-y;
    4. }
    5. else
    6. y=y-x;
    7. }
    8. return x;
```

2. Branch coverage

- In this strategy the test cases are designed to make each branch condition.
- Assuming true/ false values it is also called as edge testing.

ex- test cases are (x=3,y=3)

3. Path coverage

This strategy requires to design test cases such that all linearly independent paths in the program are executed at least once and this path can be defined in terms of CFG(Control Flow Graph)

CFG(Control Flow Graph)

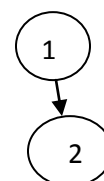
It describes the sequence in which the different instructions of a program get executed. In order to draw a CFG of a program we need to first numbering all the statements of program and diff. no statements are represented as the nodes of CFG.

- An edge from one node to another node exists if the execution of the statement representing the first node can result in the transfer of control to other node.

CFG for different kind of statements:-

1. Sequence statements

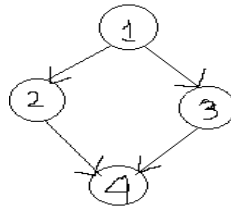
```
1. int a=5;
2. a=a+10;
```



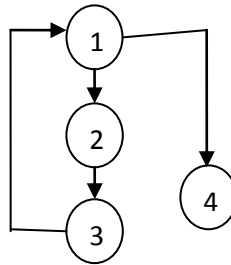
2. Selection Statement

```
1.if(a<b)
2.c=3;
3.else c=5;
```

4.c=c*c;



3. Iterative Statement



- 1.while (a>b)
2. b =b-1;
3. b =b*a;}
4. c =a+b

PATH: A path through a program is a node & edge sequence from the starting node to the termination node of the CFG.

Ex: in the above selection statement graph or CFG the paths are 1,2,4 and 1,3,4

MC CABE'S CYCLOMATIC COMPLEXITY METRIC: it provides us a practical way of determining the maximum number of linearly independent path in the program.

- It defines the upper bound on the number of independent path in program.
- Given a CFG 'G' of a program the cyclomatic complexity is defined as $V(G)$ and can be computed as **(a)** $V(G)=E-N+2$ where E =No. of edge in the graph and N =No. of Nodes in graph.
Or **(b)** $V(G)$ = total number of bounded areas +1: where bounded area means any region enclosed by a node and edge or closed path.

INTEGRATION TESTING:

- The objective of this testing is to test the module interface in order to know that there are no error in the parameter testing where one module invokes another module.
- An important factor that guides the integration plan is the module dependency graph and the following approaches are used to develop integration testing.
 - Big Bang Approach
 - Top-Down Approach
 - Bottom – Up Approach
 - Mixed Approach

Big Bang Approach:

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

- It is the simplest integration testing approach where all the modules making of a system are integrated in a single step i.e. all modules are simply put together and tested.
- It is only used for a very small system.
- The main problem with this approach is that once an error is found during integration testing is very difficult to localize the error. Hence this is very expensive.

Top Down:

- It start with the routine and one or two subordinate routine after the top level selection has been tested the immediate subroutine of the skeleton are combined with it and tested.
- The disadvantage of this approach is that in the absence of lower level routine many times it became very difficult to exercise the top level routine in the desired manner.

Bottom- Up Approach:

Here each system is tested separately and then the full system is tested.

- A subsystem might consist of many modules which communicate with each other through a well defined interface
- the main purpose of this system is to test the interface among the diff. modules making of the subsystem
- Basically this approach is used in large s/w system
- Advantage: Several disjoints of system can be tested simultaneously in a pure bottom up testing approach.

No stubs are required, only the test drivers are required.

→ Disadvantages: Complexity occurs when the system is made up of large no. of small subsystems.

MIXED APPROACH:

Follows the combination of top down and bottom up approach.

SYSTEM TESTING:

System tastings are defined to validate a fully developed system to assure that it needs its requirement.

→ There are basically 3 types of system testing

1. α -testing
2. β – testing
3. Acceptance/Rejection testing

Alpha: - It refers to the system testing carried out by the test teams within the developing orgn.

Beta: - It is done by the selected group of friendly customer.

Acceptance: - By the customer this testing is done to determine whether to accept or reject the delivery product.

Regression testing is a type of software testing that verifies that software previously developed and tested still performs correctly even after it was changed or interfaced with other software. Changes may include software enhancements, patches, configuration changes, etc. During regression testing, new software bugs or *regressions* may be uncovered. Sometimes a software change impact analysis is performed to determine what areas could be affected by the proposed changes. These areas may include functional and non-functional areas of the system.

The purpose of regression testing is to ensure that changes such as those mentioned above have not introduced new faults.^[1] One of the main reasons for regression testing is to determine whether a change in one part of the software affects other parts of the software.^[2]

Common methods of regression testing include re-running previously completed tests and checking whether program behavior has changed and whether previously fixed faults have re-emerged. Regression testing can be performed to test a system efficiently by systematically selecting the appropriate minimum set of tests needed to adequately cover a particular change.

Contrast with non-regression testing (usually validation-test for a new issue), which aims to verify whether, after introducing or updating a given software application, the change has had the intended effect.

The various regression testing techniques are:

Retest all: This technique checks all the test cases on the current program to check its integrity. Though it is expensive as it needs to re-run all the cases, it ensures that there are no errors because **Regression test selection: Unlike Retest all, This technique runs a part of the test suite (owing to the cost of retest all) if cost of selecting the part of test suite is less than retest all technique.**

Test case prioritization

Prioritize the test cases so as to increase a test suite's rate of fault detection. Test case prioritization techniques schedule test cases so that the test cases that are higher in priority are executed before the test cases that have a lesser priority.^[7]

Types of test case prioritization

- General prioritization - Prioritize test cases that will be beneficial on subsequent versions
- Version-specific prioritization - Prioritize test cases with respect to a particular version of the software.

Hybrid

This technique is a Hybrid Approach of both Regression Test Selection and Test Case Prioritization. Algorithms for the approach that have been proposed are test selection algorithm and hybrid technique proposed^[7]

Benefits and drawbacks

Regression testing is performed when changes are made to the existing functionality of the software or if there is a bug fix in the software. Regression testing can be achieved through multiple approaches; if a *test all* approach is followed it provides certainty that the changes made to the software have not affected the existing functionalities, which are unaltered, in any way.

In an agile project management process, where the software development life cycles are very short, resources are scarce and changes to the software are very frequent; Regression testing might introduce a lot of unnecessary overheads.

Typically, regression testing is carried out by automation tools, but the existing generation of regression testing tools is not equipped to handle database application. For this reason, performing a regression test on a database application could prove to be taxing as it would require a great deal of manual effort.

Moreover, performing regression testing in a software development environment which tends to use black box components from a third party can get a bit tricky as any change in the third party component may interfere with the rest of the system and performing regression testing on a third party component is difficult because it is an unknown entity

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

Reliability of software can be defined as the probability of the product working correctly over a given period of time i.e. a software product having a large number of defects are unreliable.

- The reliability of product depends not only on the number of latent errors but also on the exact location of the errors. Apart from this reliability also depends upon how the product is used.
- It is observed that reliability of software is difficult to measure.

Hardware vs. software Reliability

RELIABILITY METRICS:

It is a quantitatively measure of software product. There are six different reliability metrics which can be used to quantify the reliability of software product.

1. Rate of occurrence of Failure (ROCOF): ROCOF measures the frequency of occurrence of unexpected behavior (i.e. failure) and can be obtained by observing the behavior of software product.
2. Mean Time to Failure (MTTF): is the average time between two successive failures observed over a large number of failures. Let failures occur at the time instant t_1, t_2, \dots, t_n can be calculated as $\sum_{i=1}^n \frac{t_{i+1} - t_i}{n-1}$. It is important to note that only run time is considered in the time measurement.
3. Mean Time to Repair (MTTR): once the failure occurs some time is required to fix the error. MTTR measures the average time it takes to track the errors causing the failure and then to fix.
4. Mean Time Between Failure(MTBF):is equals to MTTF+MTTR
5. Probability of Failure on Demand (POFOD): measures the likelihood of system failing when the service request is made i.e. a POFOD of 0.001 would mean that 1 out of every 1000 service requests would result in failure.
6. Availability: of a system is measure of how likely will the system be available for use over a given period of time.

Types of failure:

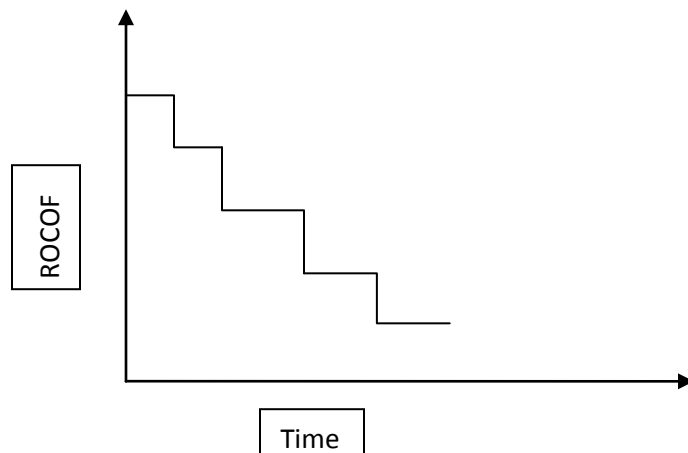
1. **Transient:** **Transient** failures occur only for certain input values while invoking a function of system.
2. **Permanent:** This type of failure occurs for all input values while invoking a function of the system.
3. **Recoverable:** When recoverable failures occur, the system recovers with or without operator intervention.
4. **Unrecoverable:** In these failures, the system may need to be restarted.
5. **Cosmetic:** This type of failure causes only minor irritations, and do not lead to incorrect results.

Reliability Growth Modeling:

Reliability growth model is a mathematical model of how software reliability improves as errors detected and repaired. A reliability growth model can be use to predict when a particular level of reliability is likely to be attained.

Types of Growth Model:

1. **Jelinski & Moranda Model:** According to them a reliability growth model is a step function model where it is assumed that the reliability increase by a constant increment each time an error is detected and repaired.



(FIG: Step function model of reliability growth)

2. **Littlewood and Verall's model:** This model allows for negative reliability growth to reflect the fact that when a repair is carried out it may introduce additional errors. It also models the fact that as errors are repaired the average improvement in reliability per repair decrease.

SOFTWARE QUALITY: quality of a product is defined in terms of its fitness of purpose. For software products, the fitness of purpose is usually interpreted in terms of satisfaction of requirements laid down in SRS document.

Quality Factors:

1. **Portability:** A software product is said to be portable, if it can be easily made to work in different operating system environments, in different machines, with other software products, etc.
2. **Usability:** A software product has good usability, if different categories of user can easily invoke the functions of product.
3. **Reusability:** a software product has good reusability, if different modules of the product can easily be reused to develop new products.
4. **Correctness.** A software product is correct if different requirements as specified in the SRS document have correctly.
5. **Maintainability:** A software product is maintainable; if errors can be easily corrected as and when they show up, new functions can be easily added to the product and the functionalities of the product can be easily modified, etc.

SOFTWARE QUALITY MANAGEMENT SYSTEM: A quality management system is the principal methodology used by organizations to ensure that the products they develop the desired quality. A quality system consist of the following

- a) **Managerial structure and individual responsibilities:** a quality system is actually the responsibility of the organization as a whole and should support the top management
- b) **Quality System Activities:** the quality system activities encompass the following
 - a. Auditing of projects
 - b. review of quality system
 - c. Development of standards, procedures, and guidelines, etc.
 - d. Production of reports for top management.

ISO 9000: ISO (International Standards Organization) is a consortium of 63 countries established to formulate and foster standardization. ISO published its 9000 series of standards in 1987

- ISO 9000 certification serves as a reference for contract between independent parties and also specifies the guidelines for maintaining a quality system.

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

- ISO 9001: this standard applies to organizations engaged in design, development, and production and serving of goods and is applicable to most software organization.
- ISO 9002: it is applicable to that organization which does not design the products but are only involved in production.
- ISO 9003: this will applies to organizations involved in installation and testing of products.

PSP (Personal Software Process):

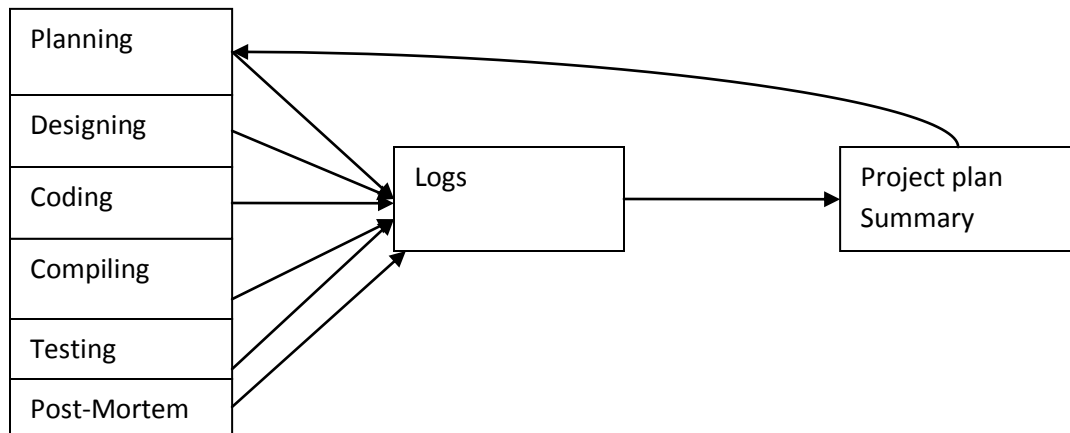


Fig: Schematic Representation of PSP

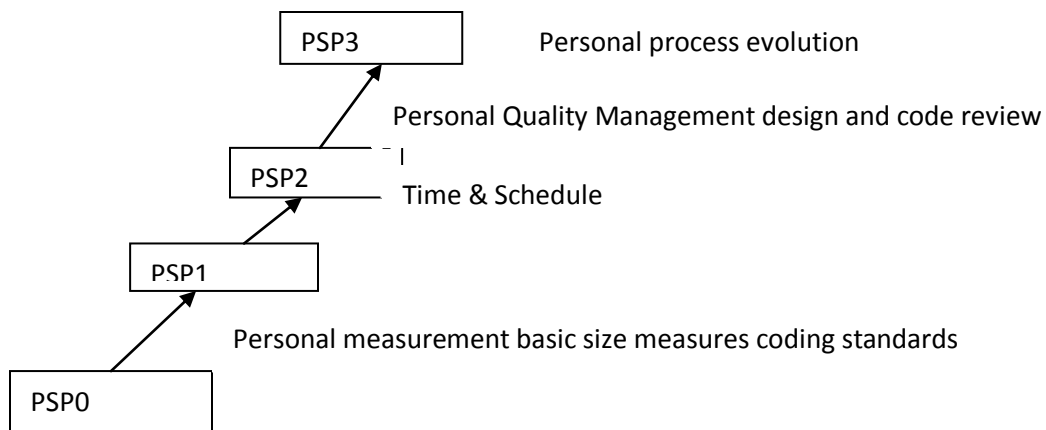


Fig: Levels of PSP

SIX SIGMA

Six Sigma is a highly disciplined process that helps us focus on developing and delivering near-perfect products and services.

Features of Six Sigma

- Six Sigma's aim is to eliminate waste and inefficiency, thereby increasing customer satisfaction by delivering what the customer is expecting.
- Six Sigma follows a structured methodology, and has defined roles for the participants.

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

- Six Sigma is a data driven methodology, and requires accurate data collection for the processes being analyzed.
- Six Sigma is about putting results on Financial Statements.
- Six Sigma is a business-driven, multi-dimensional structured approach for:
 - Improving Processes
 - Lowering Defects
 - Reducing process variability
 - Reducing costs
 - Increasing customer satisfaction
 - Increased profits

The word *Sigma* is a statistical term that measures how far a given process deviates from perfection.

The central idea behind Six Sigma: If you can measure how many "defects" you have in a process, you can systematically figure out how to eliminate them and get as close to "zero defects" as possible and specifically it means a failure rate of 3.4 parts per million or 99.9997% perfect.

Key Concepts of Six Sigma

At its core, Six Sigma revolves around a few key concepts.

- **Critical to Quality** : Attributes most important to the customer.
- **Defect** : Failing to deliver what the customer wants.
- **Process Capability** : What your process can deliver.
- **Variation** : What the customer sees and feels.
- **Stable Operations** : Ensuring consistent, predictable processes to improve what the customer sees and feels.
- **Design for Six Sigma** : Designing to meet customer needs and process capability.

Our Customers Feel the Variance, Not the Mean. So Six Sigma focuses first on reducing process variation and then on improving the process capability.

Benefits of Six Sigma

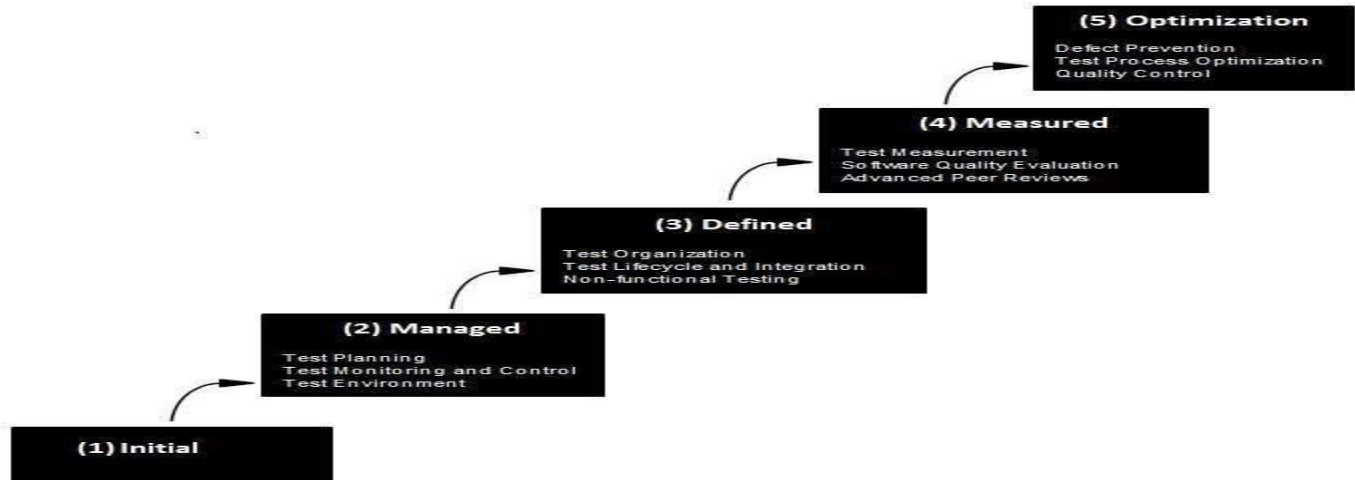
Six Sigma offers six major benefits that attract companies:

- Generates sustained success
- Sets a performance goal for everyone
- Enhances value to customers
- Accelerates the rate of improvement
- Promotes learning and cross-pollination
- Executes strategic change

Capability Maturity Model (CMM)

The Software Engineering Institute (SEI) Capability Maturity Model (CMM) specifies an increasing series of levels of a software development organization. The higher the level, the better the software development process, hence reaching each level is an expensive and time-consuming process.

Levels of CMM



- **Level One : Initial** - The software process is characterized as inconsistent, and occasionally even chaotic. Defined processes and standard practices that exist are abandoned during a crisis. Success of the organization majorly depends on an individual effort, talent, and heroics. The heroes eventually move on to other organizations taking their wealth of knowledge or lessons learnt with them.
- **Level Two: Repeatable** - This level of Software Development Organization has a basic and consistent project management processes to track cost, schedule, and functionality. The process is in place to repeat the earlier successes on projects with similar applications. Program management is a key characteristic of a level two organization.
- **Level Three: Defined** - The software process for both management and engineering activities are documented, standardized, and integrated into a standard software process for the entire organization and all projects across the organization use an approved, tailored version of the organization's standard software process for developing, testing and maintaining the application.
- **Level Four: Managed** - Management can effectively control the software development effort using precise measurements. At this level, organization set a quantitative quality goal for both software process and software maintenance. At this maturity level, the performance of processes is controlled using statistical and other quantitative techniques, and is quantitatively predictable.
- **Level Five: Optimizing** - The Key characteristic of this level is focusing on continually improving process performance through both incremental and innovative technological improvements. At this level, changes to the process are to improve the process performance and at the same time maintaining statistical probability to achieve the established quantitative process-improvement objectives.

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

Computer Aided Software Engineering

CASE stands for **Computer Aided Software Engineering**. It means development and maintenance of **software** projects with help of various automated **software** tools.

PROJECT ESTIMATE TECHNIQUE:

COCOMO (Heuristic technique)

Cost Constructive Model

- COCOMO is a heuristic based estimation technique was proposed by BOHEM in 1981.
- According him an S/w development project can be classified into three categories.
 - (I) Organic
 - (II) Semidetached
 - (III) Embedded

Organic :- A development project considered to be organic type if the project deals with developing a well understood application programmer & the size of the development team is reasonable (very small) & the team members are experienced in developing similar type of project.

Semidetached: - A development project can be considered as semidetached if the team consists of mixture of experienced, limited experience and inexperience members.

- The team members may have limited experience on related system but may be unfamiliar with some aspect of the system being developed.

Embedded: - A development project can be embedded if the S/w being developed is strongly coupled with the complex hardware on which it requiring a strict attention to the rules, procedures and all details which are already exist in the system.

- Bohem provides diff. sets of expressions to predict the efforts in units of person-month and development time from the size of the estimation given in kilo lines of source code.(KLOC)
- The COCOMO is divided in three parts.
 1. Basic COCOMO
 2. Intermediate COCOMO
 3. Complete COCOMO

Basic COCOMO: - It is an approximate estimate of the project parameters given by the expression

1. Efforts = $a_1 \times (\text{KLOC})^{a_2}$ PM
2. Tdev = $b_1 \times (\text{effort})^{b_2}$ month

a_1, a_2, b_1, b_2 are the const. for each category of S/w products.

- Tdev is the estimate time to develop the S/w products expressed in month.
- Total effort required to develop S/w product expressed in PM.

For Organic

Where $a_1 = 2.4, a_2 = 1.05$

$$\text{Effort} = 2.4 \times (\text{KLOC})^{1.05}$$

$$\text{Tdev} = 2.5 \times (\text{Effort})^{0.38}$$

where $b_1 = 2.5, b_2 = 3.38$

Semidetached

$$\text{Effort} = 3.0 \times (\text{KLOC})^{1.12}$$

$$\text{Tdev} = 2.5 \times (\text{Effort})^{0.35}$$

where $a_1=3.0, a_2= 1.12$

& $b_1= 2.5, b_2=0.38$

Embedded

$$\text{Effort} = 3.6 \times (\text{KLOC})^{1.2}$$

$$\text{Tdev} = 2.5 \times (\text{Effort})^{0.32}$$

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

Assume that the size of an organic type of S/w product has to be estimated to be 32000 lines of source code and also assume that the avg. salary of a S/w engineer is 15000/-
Determine the effort required to develop the S/w products and the nominal development time and cost of products. $32000 = 32K$

$$\text{Effort} = 2.4 \times (32)1.05 = 91 \text{ PM}$$

$$T_{\text{dev}} = 2.5 \times (91)0.38 = 14 \text{ month}$$

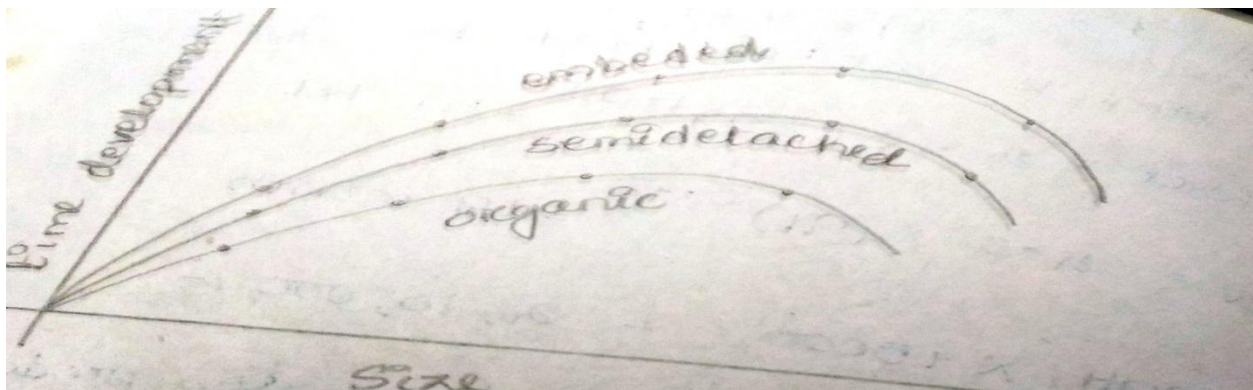
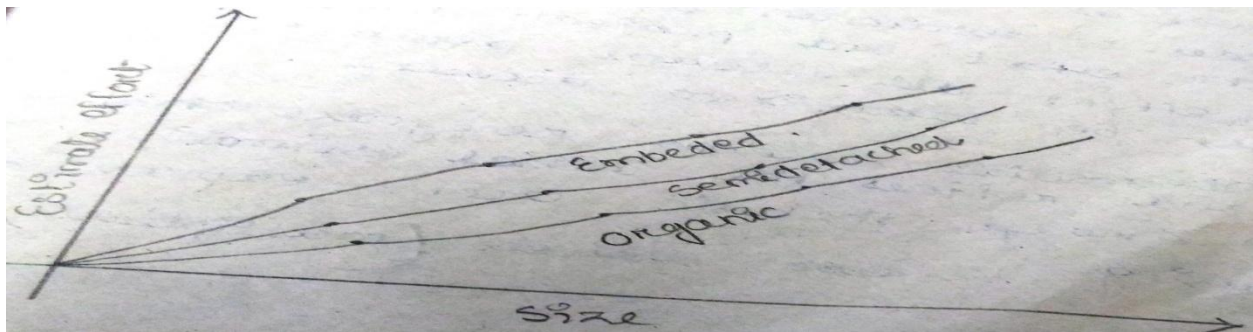
Cost:-

$$14 \text{ month} \times 15000/- = 2,10,000/-$$

Suppose you are developing an S/w product in organic mode. You have estimate the size of product 100000 lines of code complete the nominal effort and dev. time.

$$\text{Effort} = 2.4 \times (100)1.05$$

$$T_{\text{dev}} = 2.5 \times$$



Intermediate COCOMO

The basic COCOMO assumes that the important time for the development of product size. However a host of other project parameter besides the product size affects the effort required to develop the product as well as development time. Therefore in order to obtain an accurate estimation of the effort and project duration the effort all relevant parameter the effort all relevant parameter must be taken into account.

- The intermediate cocomo recognize this fact and refines the initial estimation obtain through the basic cocomo expression by using 15 cost devers (multipliers) based on various attribute. In general cost divers are classified into following items.

1. Product

The characteristics of the product, that are considered include the inherent complexity of the product, reliability and requirements of the product etc.

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

2. Computer

It includes the execution speed required, storage space etc.

3. Personnel

In includes the experienced level of personnel, program capability analysis capability etc

4. Development Environment

It captures the development facilities available to the developer. Imp Parameter sophistication of the automation tools.

COCOMO-II

➤ COCOMO-II is an updated version of COCOMO-I or basic COCOMO having the following features.

i. The model is simple and well tested.

ii. Provides 20% of cost and 70% time estimate accuracy.

➤ COCOMO-II estimate project cost derive directly from person month effort by assuming the cost is basically dependent on total physical size of the project file which is expressed in 1000 single lines of code. (KSLOC).

The formula for effort is,

$$\text{Effort} = a \times \text{KSLOC}^b$$

Where $a=3$ or 2.94

and scaling factor $b=1$

➤ COCOMO-II model defines 17 cost devices the are major influence on the cost of the project.

For e.g.:- 1 Data –database-value (high)

2 CPLX-product-complex-value (nominal)

3 time-execution time-value (nominal)

➤ It also defines 5 scaling factors that are

1. PREC-precedence-nominal

2. PMAT-product maturity level-nominal

3. Team-team cohesion-nominal

4. Flex- flexibility product-nominal

5. RESL-risk resolution – nominal

➤ And all other cost driver rather than nominal are ranges from 0.5 to 1.5

S/W matrices

➤ A S/w metrics is a measure of some property of pieces of S/w or its specification.

➤ It is a unit to define the project size.

➤ There are basically two types of S/w metric. On the basic of structure oriented.

1. Lines of code (LOC)

2. Function point (FP)

3. Feature point

1. **Lines of code:-**

➤ It is very simple among all other metrics.

➤ Here the project size is estimated by counting no. of source instruction.

➤ While counting the no. of source instruction the lines used for commenting the code & header file are ignored.

➤ In order to count the LOC at beginning of the project the project manager usually divide the program into modules & each modules onto sub module & son.

Disadvantage of LOC:-

LOC gives a numerical value of program size that can vary widely with individuals LOC measure correlates poorly with the quality & efficiency of the code i.e a code does not imply better quality or higher deficiency.

➤ LOC penalize use of higher level program language, code reuse etc.

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

2. Function point:-

- It is proposed by Albrecht in 1983.
- It can be used to easily estimate the size of an S/w product directly from the problem specification, where as in LOC the size can be accurately determine only after the product is fully developed.
- The idea behind the FP is that size of a S/w product is directly dependent o the no. of different function?
- Besides using the no. of I/P & O/P data values FP computers the size of S/w product using the B characteristics.

$$FP=UFP \times TCF$$

Where UFP=unadjusted FP

TCF=Technical complexity

$$UFP= (\text{no. of input} \times 4) + (\text{no. of o/p} \times 5) + \\ (\text{no. of enquiry} \times 4) + \text{No. of files} \times 10 \\ + (\text{no. of interface} \times 10)$$

$$TFC=0.65 + 0.01 \times DI$$

Where DI is known as degree of influence.

- TFC refines the UPF measure by considering 14 other factors such as high transaction, through put, response time, etc. each of these are assigned a value from 0 to 6.
Where 0- not present/ no influence.
- The resultant no. are summed yielding the total degree of influence.
- As the deg. of influence valy from 0.65 to 1.35.

3. Feature point metrics:-

- A major short coming of FP measure is that it does not take into account the algorithm complexity of software.
- FP is the function point metrics implicitly assure that an effort required to design and develop any functionalities of the system is same but in practical it is not true.
- Hence to overcome this problem an extension of fun. Point is called as feature point matrices.
- This metrics incorporates an extra parameter into algorithm complexity to find out the effort and development time of a software and this metric is language independent and can be easily calculate the SRS.

4. Bang metrics:-

- It is used to predict the application size based on the analysis models.
- The software engineers first evaluate the set of permissivity un sub-dividable at the analysis level.
- With evolution of this permissivity software can be defines as
1 function strong
2 Data strong

Function strong:-

The function strong metric is based on the no. of functional primitivities i.e no. of lowest level bubbles in the DFD.

- The function of primitives count is weighted according to the type of function as used in the software and the no. of data also used in the software.

Data strong:-

The data strong measure based on the no. of entitles used in ER diagram. The basic entity count is weighted according to the no. of relationship involving in each entity.

Hallstead's matrices:-

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

- It is an analytic technique to measure the size development effort and development cost of software.
- This person used or reused some primitives programming parameter to develop the expression for the overall program length, potential minimum volume and actual value of the program, effort, time development

For a given program UT

n_1 = no. of unique operators used in a program.

n_2 = no. of unique operators used in a program.

N_1 = total no. of operators used in a program.

N_2 = total no. of operators used in a program

Operators:-

- All the operators used here is same as ANSI but we can also take the following statement as operator:-

Switch, goto, while, for, break, continues and a funⁿ name in a function call for example:-

int func (int a, int b)

Fun c (a, b)

Operators are:-

() operands, a, b

How to calculate the length of a program?

Length is defined as total no. of use of all operators and operands

$$N = n_1 + n_2$$

Program vocabulary:-

$$N = n_1 + n_2$$

Volume:-

Volume of a program is defines as minimum no. of bits needed to encode a program.

$$V = n \log_2 n$$

Potential volume (V) is defined as the volume of the most succulent program in which a problem can be code.

$$V^* = (2 + n_2) \log_2 (2 + n_2)$$

Program level (l):-

$$L = V^* / v$$

$$\text{Effort } E = V / L$$

$$E = \frac{v^2}{v^*}$$

Time to development $T_{dev} = \frac{E}{S}$

Where S = speed of metal discrimination

S = 18 (average value) medium size

Estimation of length $N = n_1 \log_2 n_1 + n_2 \log_2 n_2$

For example:-

Program Level (e):-

$$E = V^* / V$$

$$\text{Effort } E = V / e$$

$$E = \frac{V^2}{V}$$

$$\text{Time to development, } T_{dev} = \frac{E}{S}$$

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

Where S = speed @ mental discrimination

S = 18 (average value) // medium size estimation on length, N=n

For example

```
main ( )
```

```
{
```

```
int a, b, c, avg,
```

```
scan ( l X d X d X d X
```

```
avg = ( a + b + c ) / 3
```

```
print (avg = d", avg)
```

Unique operators :-

main, C, {, int, scan, \$ = +, l, print

Operands (Unique) :-

A, b, c, \$a, \$b, \$c, a+b+c, avg, 3, d, X d X d"

"avg = ya"

N2 = 11

$N = n \log_2 n_1 + n_2 \log_2 n_2$

= 12 log 12 + log 11

= 12.95 + 11.45 = 24.40

Design Matrix :-

1. High Level

(a) Structure Complexity.

(b) Data Complexity

(c) System Complexity

2. Component Level

(a) Cohesion Metric

(b) Coupling Metric

(c) Complexity

High Level :-

(1) Structure Complexity :-

Structure complexity on a module "I" is given as $S(i) = F_{out}(i)$

Where $F_{out} = F_{in}$ out i.e. the no. of modules immediately directly called

(2) Data Complexity :-

The data complexity on a module 'I' is given as.

$$D(I) = \frac{V(i)}{[f_x + (i) + l]}$$

Where $V(i)$ = no. of inputs and outputs passed to and from (i)

(3) System Complexity :-

The system complexity of module (i) is given as $C(i) = S(i) + D(i)$

Component Level :-

(1) Cohesion :-

Data slice Data values that affect the module location at which the backward trace began.

Data taken The variables defined for that module

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

Glue taken The set of token lying on multiple data slice.

Super glue The set of tokens present in all slice.

(2) Coupling :-

The coupling matrice $mc = \frac{k}{m}$

Where k=constant value =1

$M=di + (a + ci) + di + (h + Co) + gd + (c + gc) + w + r$

Di = Input data parameter & Ci = input control parameter

Do = Output data parameter & Co = Output control parameter

Gd = Global data variable & Gc = Global data variable

W = Fan in & R = Fan out

(3) Complexity :-

The complexity motive can be calculated by using control flow graph (CFG)

WEB ENGINEERING:-

- Web Engineering (WEBE) is a process tht is used to create high quality web application.
- Web Engineering is not a perfect clone for software engineering but it borrows many software engineers fundamental concepts and principles.
- In addition the web engineering process emphasizes on similar technical and management activity.

Importance of WEBE :-

As webapps become increasingly integrated in business strategies for a small & large companies.

Eg :- Commence, the need to build reliable, useable & adaptable systems grows in importance, that's why a disciplined approach to web apps development in necessary.

ATTRIBUTE OF A WEB BASED SYSTEM & APPLICATION

(1) n/w intensiveness

- A web application resides on a n/w & must serve the needs of the client.
- A web apps may reside on the internet, alternatively an application may be placed on the intranet or an externet.

(2) Concurrency

- A large no. of users may access the web application at a time.

(3) Unpredictable load

- No. of users of a web application may vary by orders of magnitude from day to day.
i.e. 100 user may show up on Monday & 10000 users on Thursday.

(4) Performance

- If a web application user must wait too long time & he/she may go to elsewhere.

(4) Availability

- Web application must available for 24 X 7 X 365

(5) Data Driven

- Primary function of many web application to be used hypermedia or hyperlink to present text, graphics, audio, video & etc. and also for common use access information that is stored in database.

(6) Content Sensitivity

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

- The quality & aesthetic nature of content remains as an important determinant of the quality of web application.
- (7) Continuous Evaluation
 - Conventional application S/w that evaluate over a series of planned chronological order but in case of web application continuous evaluation of that application is needed.
- (8) Security
 - In order to protect sensitivity data we need to provide a secure mode of data transmission on our web application for that a strong security measures must be implemented.
- (9) Aesthetics
 - An undeniable part of appeal of a web application is its look & feel.
- (10) Immediacy
 - Need to get S/w to market quickly.

APPLICATION CATEGORIES ENCOUNTERED ON WEB APPLICATION :

- (1) Informational
- (2) Download
- (3) Customize
- (4) Interaction
- (5) User i/p
- (6) Transaction oriented
- (7) Service oriented
- (8) Portal
- (9) Database access
- (10) Data warehouse

PLANNING

The project plan for web application increment is created.

The plan consist of task definition & timeline schedule for the time period projected for the development of web application.

MODELING

The conventional S/w engineering analysis & designing tasks are adapted to web application development.

CONSTRUCTION

Web engineering tools & technology are applied to construct the web application that has been modeled.

Once the web application has been constructed a series of test has been conducted. i.e. alpha, beta & acceptance test.

DEPLOYMENT

The web application is configure for its operational environment & delivered to the end users & then an evaluation period, commences & the feedback is presented to the web engineering team.

CUSTOMER COMMUNICATION

Within the web engineering process customer communication is characterized two major tasks.

1. Business analysis: It defines the business & organizational contest for the web application.
2. Formulation: It is the requirement gathering activity involving all stake holders.

WEB DESIGN

Characteristics of web design:

- (1) Simplicity
 - (2) Consistency
 - (3) Identity
 - (4) Robustness
 - (5) Navigation
 - (6) Visual appeal
 - (7) Complexity
- Interface design describes the structural & organizational of the user interface which includes the represent the screen layout.

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

- Aesthetics design also called as graphical design. This describes the look & feel of the web application.
- The content design defines the layout & structure & outline for all web application content.
- It represents the navigation flow between content objects & for all web application. This is finding path from one application to another.
- Architectural design identifies all the overall hyper media or hyper link of a structure.
- In component design it gives the detail processing logic required for web designing.

OBJECT ORIENTED METRIC

- (1) Size (2) Complexity (3) Coupling(4) Sufficiency (5) Completeness (6) Cohesion (7) Primitiveness
(8) Similarity (9) Volatility

CLASS ORIENTED METRICS

CK MATRICS SUITE:

- CK metrics was proposed by chidamber and kumerer.
- The author have proposed 6 class base design metrics.

(1) Weighted method per class (WMC):

Assume n method of complexity C1,C2,.....Cn are detined for a class c. The specific complexity that is choosen should be normalized so that the nominal complexity for a method takes on a value 1.

(2) Depth of inheritance tree:

This metric has the maximum length from the node to the root.

(3) Number of children (NOC):

The subclass that are immediately subordinating to a class in the classes hierarehy are turned its children.

(4) Coupling between object classes :

The CRC model may be used to determine the value for CBO is the no. of collaboration listed for a class on its CRC index card.

- As CBO increases it is likely that the reusability of a class will decrease.

(5) Response for a class : (RFC) :

The response set of a class is a set of methods that can be potentially executed in response to a message received by an object of that class.

- As RFC increases the effort require for testing is also increases.bcz the test sequence grows.

(6) Lack of Cohesion in methods (LCOM) :

- Each methods within a class access one or more attributes or instance. The LCOM is the no. of methods that access one or more of the same attribute.

- If no methods access the same attribute then LCOM = 0

- This metric is proposed by Harrison, counsell & nithi to provide quantitative indicators for object oriented design.

MOOD METRICS SUITE:

Method inheritance factor (MIF):

- The degree to which the class architecture of an object oriented system makes use of inheritance for both method & attribute is defined as $MIF = \frac{\sum_{i=1}^{TC} mi(Ci)}{\sum_{i=1}^{tc} Ma(Ci)}$

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

Where summation occurs over $l = 1$ to TC

- TC is defined as the total no. of classes in the architecture & C_i is the class within the architecture.
- Aim of $C_i = (M_a) C_i = M_d(C_i) + M_i(C_i)$
 $M_a(C_i)$ is the no. of methods that can be involved in association with C_i .
- And M_d of C_i is no. of methods declared in the C_i .
- M_i of C_i . Is the no. of methods inherits in C_i .

Coupling Factor:

In coupling factor (CF) is calculated $CF = \frac{\sum_{i=1}^{TC} \sum_{j=1}^{TC} \text{client}(C_i, C_j)}{TC^2 - TC}$

- The function is client = 1, if and only if the relationship exist between the client class & server class.
- If the relationship does not exist then client function = 0

Software Maintenance

Necessity of software maintenance:

Software maintenance is becoming an important activity of a large number of software organizations. This is no surprise, given the rate of hardware obsolescence, the immortality of a software product per se, and the demand of the user community to see the existing software products run on newer platforms, run in newer environments, and/or with enhanced features. When the hardware platform is changed, and a software product performs some low-level functions, maintenance is necessary. Also, whenever the support environment of a software product changes, the software product requires rework to cope up with the newer interface. For instance, a software product may need to be maintained when the operating system changes. Thus, every software product continues to evolve after its development through maintenance efforts. Therefore it can be stated that software maintenance is needed to correct errors, enhance features, port the software to new platforms, etc.

Types of software maintenance there are basically three types of software maintenance. These are:

- Corrective: Corrective maintenance of a software product is necessary to rectify the bugs observed while the system is in use.
- Adaptive: A software product might need maintenance when the customers need the product to run on new platforms, on new operating systems, or when they need the product to interface with new hardware or software.
- Perfective: A software product needs maintenance to support the new features that users want it to support, to change different functionalities of the system according to customer demands, or to enhance the performance of the system.

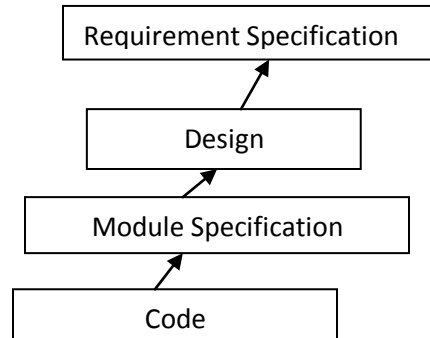
Software reverse engineering

Software reverse engineering is the process of recovering the design and the requirements specification of a product from an analysis of its code. The purpose of reverse engineering is to facilitate maintenance work by improving the understandability of a system and to produce the necessary documents for a legacy system. Reverse engineering is becoming important, since legacy software products lack proper documentation, and are highly unstructured. Even well-designed products become legacy software as their structure degrades through a series of maintenance efforts. The first stage of reverse engineering usually focuses on carrying out cosmetic changes to the code to improve its readability, structure, and understandability, without changing of its functionalities. A process model for reverse engineering has been shown in fig. A program can be reformatted using any of the several available prettyprinter

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

programs which layout the program neatly. Many legacy software products with complex control structure and unthoughtful variable names are difficult to comprehend. Assigning meaningful variable names is important because meaningful variable names are the most helpful thing in code documentation. All variables, data structures, and functions should be assigned



(FIG: A Process Model for Reverse Engineering)

meaningful names wherever possible. Nested conditionals in the program can be replaced by simpler conditional statements or whenever appropriate by case statements.

Factors on which software maintenance activities depend

The activities involved in a software maintenance project are not unique and depend on several factors such as:

- The extent of modification to the product required
- The resources available to the maintenance team
- The conditions of the existing product (e.g., how structured it is, how well documented it is, etc.)
- The expected project risks, etc. When the changes needed to a software product are minor and straightforward, the code can be directly modified and the changes appropriately reflected in all the documents

Software maintenance process models: Two broad categories of process models for software maintenance can be proposed. The first model is preferred for projects involving small reworks where the code is changed directly and the changes are reflected in the relevant documents later. This maintenance process is graphically presented in fig. In this approach, the project starts by gathering the requirements for changes. The requirements are next analyzed to formulate the strategies to be adopted for code change. At this stage, the association of at least a few members of the original development team goes a long way in reducing the cycle time, especially for projects involving unstructured and inadequately documented code. The availability of a working old system to the maintenance engineers at the maintenance site greatly facilitates the task of the maintenance team as they get a good insight into the working of the old system and also can compare the working of their modified system with the old system. Also, debugging of the reengineered system becomes easier as the program traces of both the systems can be compared to localize the bugs.

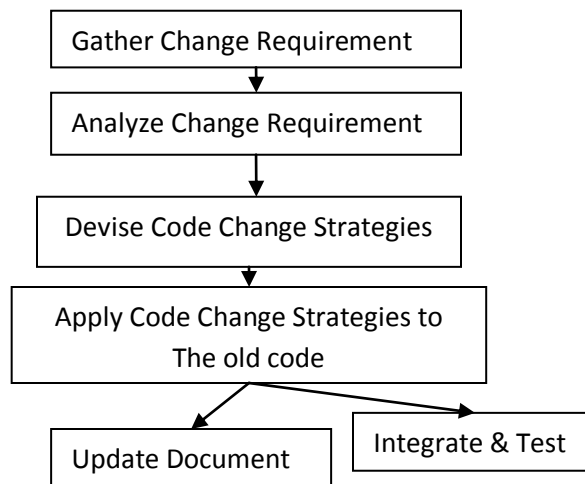


Fig : Maintenance Process Model

Service-Oriented Architecture (SOA)

A **service-oriented architecture (SOA)** is a style of software design where services are provided to the other components by application components, through a communication protocol over a network. The basic principles of service oriented architecture are independent of vendors, products and technologies.^[1] A service is a discrete unit of functionality that can be accessed remotely and acted upon and updated independently, such as retrieving a credit card statement online.

A service has four properties according to one of many definitions of SOA.

1. It logically represents a business activity with a specified outcome.
2. It is self-contained.
3. It is a black box for its consumers.
4. It may consist of other underlying services

SOA separates functions into distinct units, or services, which developers make accessible over a network in order to allow users to combine and reuse them in the production of applications. These services and their corresponding consumers communicate with each other by passing data in a well-defined, shared format, or by coordinating an activity between two or more services.

A manifesto was published for service-oriented architecture in October, 2009. This came up with six core values which are listed as follows.

1. **Business value** is given more importance than technical strategy.
2. **Strategic goals** is given more importance than project-specific benefits.
3. **Intrinsic inter-operability** is given more importance than custom integration.
4. **Shared services** is given more importance than specific-purpose implementations.
5. **Flexibility** is given more importance than optimization.
6. **Evolutionary refinement** is given more importance than pursuit of initial perfection

Elements of SOA:

GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI BHAWANIPATNA

PREPARED BY MR.GOPAL BEHERA, ASST. PROFESSOR, CSE

