Software Engineering .

Software - It is a collection of programs that when executed provides desired functions & performance Ex Ms word, Bank Management Sys. etc.

Engineering - The process of using knowledge and principles to design, build and analyze objects.

Defn - " Software engineering is the application of engineering principles to the design, development, maintenance and implementation of software ."

Software Product - Software when made for a specific requirement is called Software product.

Software Process - It is the set of activities and associated result that produce a software product

Or

The process that deals with the technical and management issues of software development is called a software process.

Diadvatages -

① Only system that can be modularized can be built using RAD.

② Requires highly skilled developers / designers.

③ Inapplicable to cheaper projects as cost of modelling & automated code generation is high.

④ For larger system, RAD model require sufficient human resources.

⑤ The customer and developer ~~showed~~ should be active because the laziness of any one may lead to failure.

When to use RAD model -

- When there is a model to create a system that can be modularized in 2-3 months of time.
- When there is high availibility of designers for modeling.
- When budget is high enough to afford their cost along with the cost of automated code generating tools.

- Members of the mangers group can enter or approve a requests but cannot delete requests.
- Members of the Administrators group cannot enter or approve requests but can delete requests.

How to identify the functional requirements ?

It can be identified either from an informal problem description of the problem a conceptual understanding of the problem.

- For this the different types of users of the system might be identified. and then the requirements from user's prespective.

Non- Functional Requirements.

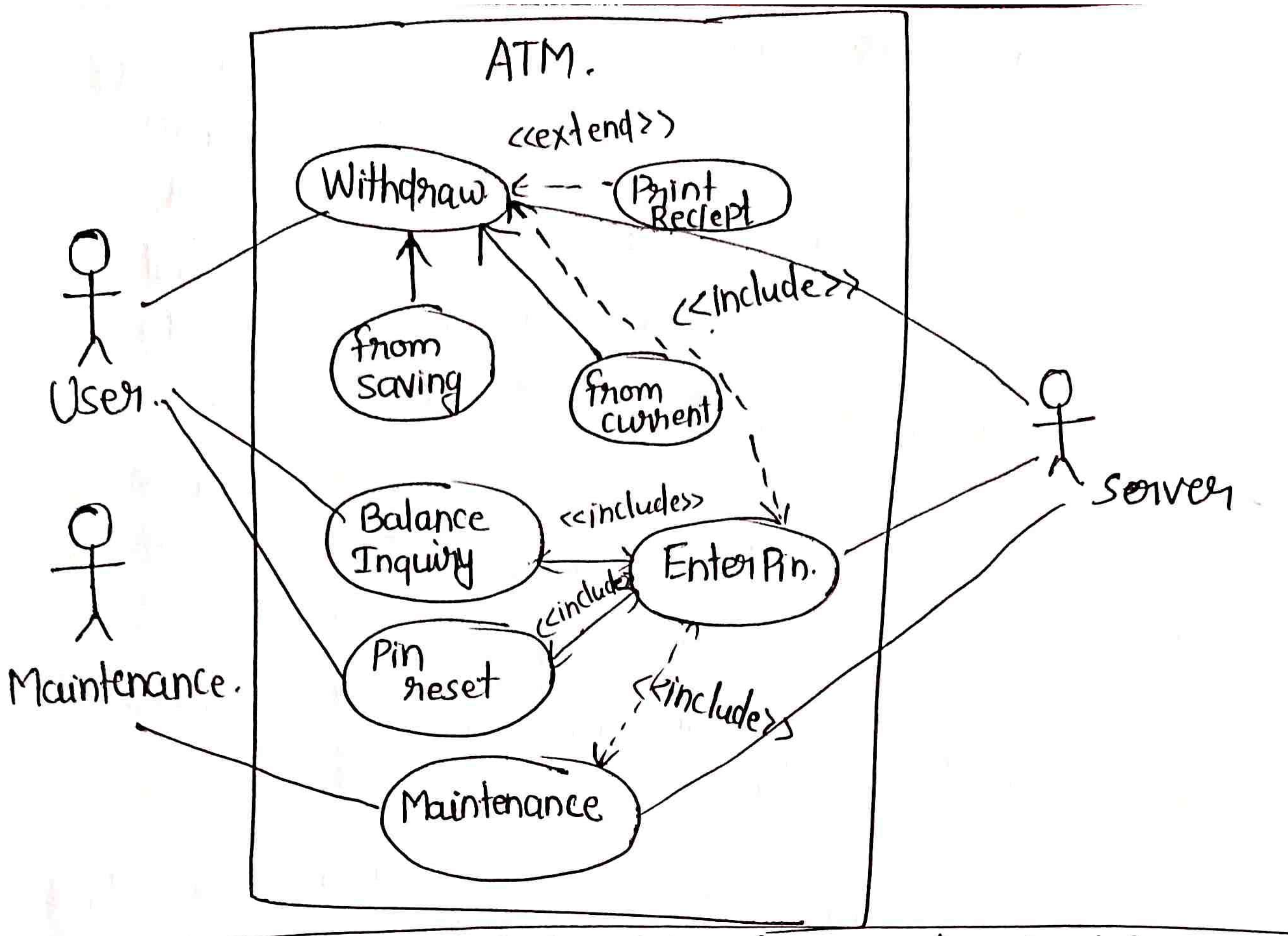
Requirement, which are not related to functional aspect of s/w, fall into this category.

They are implicit or expected characteristics of the software, which users make assumption of.

- Non functional requirements mostly define the overall attributes of the resulting system.

It defines the constraints on how (functional) requirements are to be met.

Eg:- Response time must be + second per screen.



Object oriented models (diagram) to define Classes of objects & their states

CLASS DIAGRAM. - A class diagram is the Unified Modeling language (UML) is a type of static structure ~~diagram~~ + diagram that describes the structure System by the System's classes, their attributes, operations/ methods and the relationship among objects.

- The main purpose of class diagram is to model/ Analysis/ design/ of the static view of an application. ~~[illegible]~~
- Class diagram are the only diagram which can be directly mapped with object oriented language and thus widely used at the time of s/w construction.

Bottom up Integration - It is also based on incremental approach & starts from lower level modules, moving upwards to the higher level modules.

Again the higher level modules might not have been developed by the time lower modules are tested. So in those cases Drivers are used. These drivers simulate the functionality of higher level modules in order to test lower level modules.

Advantage - We don't have to wait for all the modules to get developed before start testing

In bottom-up Integration, several disjoint subsystems can be tested simultaneously.

But complexity arises when there are large numbers of small ~~sy~~ subsystems.

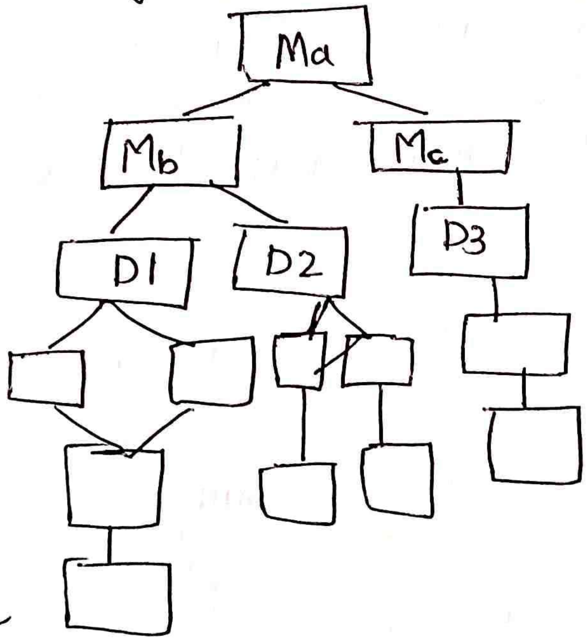


Fig - Bottom up Integration

System Testing - System testing is the level of testing where the complete integrated application is tested as a whole

- It aims at determining if the application conforms to its bussiness requirements.

-

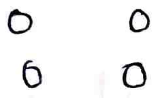
(9) Object oriented design programming done con-currently with otherphases	(9) Structured design programming usually left until end phases.
(10) It is suitable for most business applications, game development projects, which are expected to customize or extended.	(10) It is suitable for a real time system, embedded system & projects when objects are not the most useful level of abstraction.
(11) Class diagram, Sequence diagram, state chart diagram & use cases are used	(11) DFD & ER diagram model the data.
(12) Message passing is used	(12) Function call is used.
(13) Object oriented approach uses iterative & bottom up processes rather than step by step. All life cycle activities are performed in each project iteration	(13) structural approach uses traditional, waterfall, incrementals model of SDLC. [Planning - Analysis - Design - Implement - maintenance].
(14) Requires more technically skilled analysis & designers	(14) Requires less technically skilled as compared to object oriented approach
(15) Suitable for in house development	(15) more suitable for off shoring.
(16) Not so-clear transition from design to implementation	(16) clear transition from design to implementation.

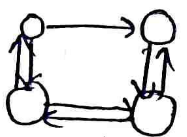
- Each ~~mem~~ member function of the object should perform unique action/ operation / function.

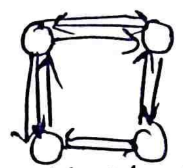
Conclusion/ Summary - cohesion measures relatedness of the elements encapsulated in one module.
- Each element in a module should be necessary & essential part of one & only task.

COUPLING

Coupling is the measure of the degree of interdependece between modules.


uncouple modules
- no dependencies


Loosely coupled
- some dependency


Highly coupled
many dependency.

- Coupling is a measure of the extent of information interexchange between modules.
- Tight coupling implies large dependence on the structure of one module by another because higher number of connections, more paths along which errors can extend into other parts of the program.
- Tight coupling makes modifying parts of the system difficult e.g. ~~mel~~ modifying a component affects all the components to which the component is connected.

∴ Modules should have low coupling

Loose coupling makes it possible to.
- Understand one class without reading others.
- change one class without affecting others.
- Thus improves as maintainability.