5th May 2017.

Software Engineering:

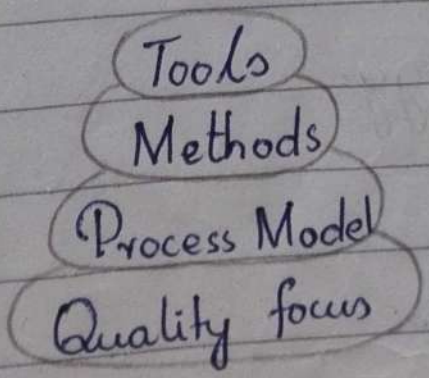
The application of systematic, disciplined, quantifiable approach to the development, operation and maintenance of SW and study of these approaches that is application of engineering to SW.

Common Issues:

→ doesn't fulfil need of customer.
→ hard to extend and improve.
→ Bad documentation.
→ Bad quality
→ More time and cost than expected

SE problem solving approach:

→ Analysis
→ Synthesis
→ SE layered technology.



② trapdoor → a programmer leaves a loop hole for the purpose of extension and improvement. They are hidden and only visible to programmer.

* SQL injections → are used to make changes in the database. They are used to find trapdoors.

③ If not properly documented then others can't continue that project.

④ features are not according to the user requirement.

⑤ Time of project & cost

8 May 2017.

extra

Problem Solving Approach:

Analysis:

Synthesis:

Layered technology:

Tools

Methods

Process Models

Quality focus.

extra

Generic View:

Definition phase:
- System and information engg.
- SW project plan.
- Requirement Analysis.

Development:
- SW design.
- Code Generation.
- SW Testing.

Support phase:
- Correction
- Adaptation
- Enhancement
- Prevention.

Q: Problem Solving Approach:

S.E is problem solving approach. People find out the problems around them and then bring a solution of that problem like Uber and Careem.

Analysis:

→ In the analysis of S.E problem:
→ finds out the problem.
→ Understand the problem.
→ Divide it into chunks so that its solution will become easy. For example

On a student portal system. We can divide this project in several chunks:
→ Student's reviews
→ Teachers reviews
→ Advisers reviews
→ Login pannel etc.

Synthesis:
We then combine the smaller chunks to form larger chunk so as to create / design interfaces and to form data bases. In short, we deeply analyze the problem during synthesis.

Layered Technology: (Diagram)
If SE is problem solving approach then SE is called Layered Technology. They layers are interdependent on each other

→ Quality focus:
You must know that how to enhance the quality of project so as to increase its market value. For example, we have many similar game like temple run but some of them might lack in

quality which will reduce customer's satisfaction.

→ Process Models:
For the details of the product, we use different process models according to the product requirement

→ Methods: Method contain a broad array of tasks that include communication req. analysis design modeling, program construction tasks and support
Different methods are used to work on the product

→ Tools: provide automated or semi-automated support for the process and methods
Like google form for the requirement, Excel to enter the requirement. These can be different tools for the project.

10 May 2017

Q. Generic View:
When you work to build a product or system, it's important go through a series of predictable steps that helps you to create a timely, high quality result.

01. Definition phase:
This phase includes:
→ System and information engg.
→ SW project plan
→ Requirement Analysis.

10 May

Definition process tell us what to do. We acquire system information, plan the project that which step should be taken first. The gathered information is then analyzed and prepare a SRS document in this phase.

02. Development phase:

It includes the following steps:

→ SW design: in which we give the graphical representation of our software project.

→ Code Generation

→ Testing:

Once the code is written. We then look for the bugs and errors in the project.

03. Support phase:

Then finally we do the postmortem of the project to improve its effectiveness This phase includes:

→ Correction of the errors and bugs.

→ Adaptation:

Zeeshan
Saba Sadaf Kashaf Nimra

To adopt or embed the already existing features in your software

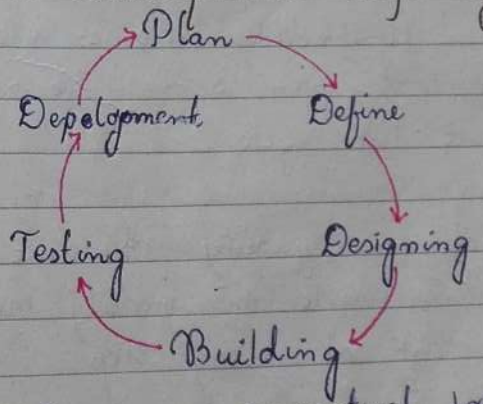
→ Updation / Enhancement:

Keeping in view, the future requirements or extensions, the software should be made powerful as to improve the previous features or add some new according to the user needs.

Process Models: that in which manner it is to be executed.

Q: SDLC:

(Software Development Life Cycle)



SDLC is a conceptual framework to be used as a guide in making a diagnosis, understanding a developmental process and to produce a continuous process. SDLC models are made to mana

ge the level of complexity. It describes two types of activities or SDLC models.

01. Traditional or conventional model (activity):

It is a step-by-step approach. The activities of one phase must be completed before moving to the next phase like waterfall model

e.g Quality Management whose scope lies on Development just.

02. Umbrella activity:

Umbrella activities are non-SDLC activities that span across the entire software development life cycle.

e.g:

01. Software quality assurance

To ensure the quality measurement of each part at every step.

e.g: 02. Risk management:

A series of steps that help a software team to manage uncertainity. It's good to identify it, estimate its impact and establish a contingency plan.

Plan:

Plan includes the requirement gathering and analysis to check how much idea can put into the action. This is basically the brainstorming phase which include cost, bugjet estimates etc.

02. Define:

This is the second phase of SDLC where the entire system is defined in detail, The system is divided into smaller chunks to make it easier for developers, designers, testers to work on it. SRS is developed (Software Requirement Specification)

03. Designing:

On this phase, designer get the basic idea of designing the software design of front end and back end both. Flow diagrams are made.

04. Coding/ Building:

On this phase, code of software is generated by the coders according to the prepared design

05. Testing:

The software is then tested by using different tools and techniques. The different errors and bugs are removed.

06. Development:

This is the final phase of SDLC. The software is deployed on different system. In this stage, if this software runs smoothly on different systems without any flaw then its ready to launch.

Prescriptive Process Model:

Defines a distinct set of activities, actions, tasks, milestones and work products that are required to engineer high-quality software.

The activities may be:

* Linear
* Incremental
* Evolutionary.

* Linear Model:

Also known as Waterfall model. Workflow is in sequential manner. It is a step-by-step approach. Activity of one phase is completed before going to the next.

* Incremental Model:

Is useful when multiple independent deliveries are identified. It is iterative in nature and is staggered between increments. It focuses on the delivery of an operational product with each increment.

* Evolutionary Model:

are iterative. They are characterized in a manner that enables software engineers to develop increasingly more complete versions of the software.

Importance :-

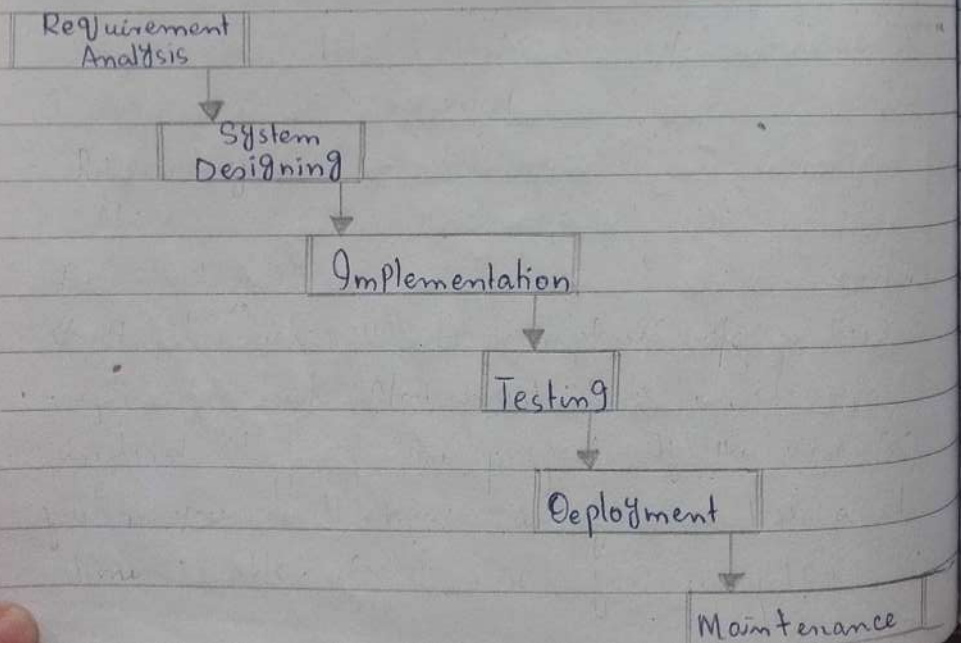
SDLC model follows traditional approach. They manage the level of complexity of your project. They are simplest models and due to this reason many people or developers still follow SDLC models rather than Agile.

15 May 2017 Classical Model

Waterfall Model: (Eg of linear model)

It is the first SDLC model in which once a phase of development is completed, the development proceeds to the next phase. It can not turn back. It has following specialities:

- Linear, Sequential Life Cycle Model.
- Simple to understand and use.
- Each phase must be completed before going to other.
- Overlapping is not allowed, can't execute two phases parallel.
- O/p of one is an input of other.



Applications:

→ When we have clear, understood and fixed requirement.
→ It can be used for short project.
→ If you have a full idea of technology, that it will not change for coming few years.
→ When you have plenty of resources and expeties.

Advantages:

→ It is simplest of all models
→ It involves linear approach.
→ Easy to understand and handle.
→ Its milestones are clear (means outputs)
→ Rigdless.

Disadvantages:

→ Can't go back to the previous phase which is the major drawback.
→ You must have a clear understanding of the project. If the waterfall model fails you will have a finance loss and time loss.
→ Can't use this model for complex

projects and ongoing projects.
→ Greater risk involved
→ All units are integrated together, can't integrate two or three units first – Big Bang Incrementation
→ No feedbacks are taken.

17 May 2017

Prototyping Model:

The prototyping model is a system development method (SDM) in w/c a prototype (an early approximation of a final system) is built to understand customer requirement at an early stage and helps in getting feedback from customer. It is used when all the requirements are not known.

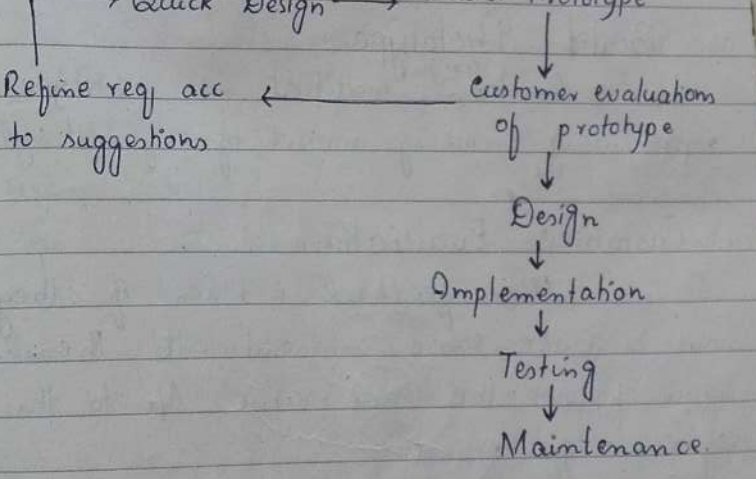
Advantages:

→ Increased user involvement in the product even before its implementation.
→ As the working model is developed, the users get clear understanding.
→ Reduces time and cost as defects can be detected earlier.
→ Quicker user feedback is available.
→ Missing funktionality can be identified.

Disadvantages:

→ The effort invested in building prototypes may be too much if it is not mentioned properly.
→ Users may get confused in prototypes and actual system

Representation



2 types of prototype:
On horizontal dimension, we talk about interface. On vertical dimension, we talk about working flow / internal functionality.

01. Requirement Analysis:

02. Quick Design:
When requirements are know a quick or preliminary design is made. It is not detailed design. Shows only the important aspects of system and helps in building a prototype.

03. Build Prototype:
Quick design is modified into prototype representing working model of required system.

04. Customer Evaluation:
User's feedback is taken if they want to make some ammendments then again follows the same cycle. A/c to the refine requirements.

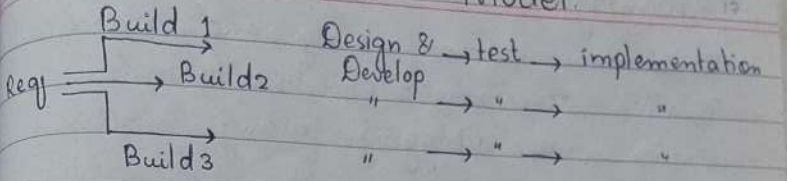
05. Design:

06. Otherwise if user accepts the

07 prototype then design of the project

08. is made.

Iterative Model



It focus on an initial, simplified implementation which then progressively gains more complexity.

- At each iteration, design modifications are made and new functional capabilities are added.
- The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portion of time (incremental)
- The whole requirement in divided into various builds. During each iteration, the dev model goes through req, design, implementation & testing phase until the complete system is ready as per the requirement.
- When we have large no. of requirements

Advantages:

- Results are obtained early and periodically
- Parallel development can be planned.
- Testing and debugging during smaller iteration is easy.
- It supports changing requirements.
- Better suited for large and mission-critical projects.

Disadvantages:

- More resources may be required.
- Not suitable for smaller projects.
- Management complexity is more.

Spiral Model:

The spiral model is similar to the incremental model and a risk-driven process. A software model repeatedly passes through these spirals. It is used when:

- When risk evaluation is important.
- For medium to high-risk projects
- When users are unsure of their needs
- Requirements are complex

Planning:

Requirements are gathered during this phase like SRS document.

Risk Analysis:

In risk Analysis, a process is undertaken to identify risk and if any risk is found then alternate solutions are suggested and implemented.

Engineering:

In this phase software is developed along with testing at the end.

Evaluation:

This phase allows the customer to evaluate the output of the project to date before the project continues to next spiral.

Customer Communication:

If customer agrees, the project is implemented otherwise it goes to next spiral.

Advantages:

- High amount of risk analysis.
- Good for large and mission-critical projects.
- Strong approval & documentation control
- Additional functionality can be added at a later date.

Disadvantages:

- Can be costly model to use.
- Risk analysis require highly specific expertise.
- Project's success is highly dependent on the risk analysis phase.
- Doesn't work well for smaller projects.

It was difficult to convince the people that spiral controllable, so the people followed traditional cycles Win-Win Spiral. On w/e customer feedback is taken in every step.

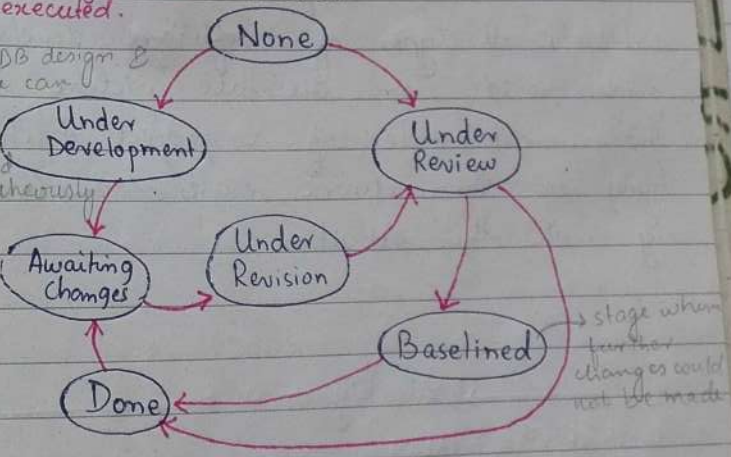
Q: Identify Risks in Library Management System.

→ Portal Crash
→ Security issue.
→ If the system is not user-friendly
→ Not proper requirement gathering.

26 May 2017.

Concurrent Development Model:

parallel stages can be executed.
e.g.: DB design & interface can be developed simultaneously.



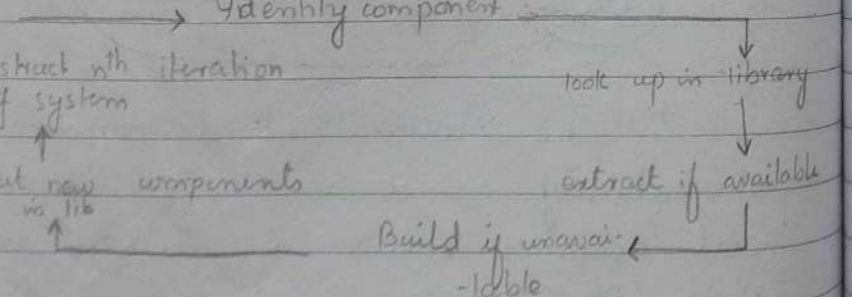
The concurrent development schematically as a series of major technical activities, tasks and their associated states. All activities exist concurrently but reside in different states.

Concurrency is achieved in two ways:

01. System & component activities occur simultaneously and can be modeled using the state-oriented approach.

02. a typical client/server application is implemented with many components, each of which can be designed and realized concurrently.

Concurrent process model is applicable to all types of software development and provides an accurate picture of the current state of a project. Each activity on the network exists simultaneously with other activities.



Agile Methadologies:

Group of software development methodologies based on iteration development where requirements and solutions evolve through collaboration of self-organizing cross-functional teams. It refers to any development process i.e is aligned with Agile manifestoes.

Manifestoes:

- Individual and iterations over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

Agile Software development principles:

01. Customer's satisfaction by early and continuous delivery of valuable software

02. Welcome changing requirement even in late development.
03. Working software is delivered frequently in weeks rather than months.
04. Close daily corporation b/w business people and developers.
05. Projects are built around the motivated individuals.
06. Face to face conversation is best form of communication (co-location).
07. Working software is principle measure of progress.
08. Sustainable development able to maintain constant phase.
09. Continuous attention to technical excellence and good design.
10. Simplicity art of maximizing amount of work not done is essential.
11. Best architecture requirements and design emerge from self-organizing teams.
12. Regularly team reflects on how to become more effective and adjust accordingly.

29 May 2017.

Agile Software Development Methods:

* Dynamic System Development Methods (DSDM)
  Full dev Cycle
* Extreme Programming (XP)
  Focus on Practices
* Feature Driven Development (FDD)
  Support activities for req specification & development
* Lean Software Development

* Kanban } Manage workflow.
* Scrum }

Extreme Programming (XP):

intended to improve quality and responsiveness to changing requirements.

XP Activities:

In XP, there are 4 basic activities:

Coding:

Coding is done while using pair programming technique in which project team work together

Testing:

Testing is done to check errors or bugs in the code when done with the coding.

Listening:

In pair programming, team collaboration is important. You listen to every single member of the team.

Designing:

Designing is done so as to get clear understanding about what are you coding and testing.

refactorization → To simplify or restructure your code.

There are 12 XP practices grouped into 4 areas:

01. Fine scale feedback:

* Pair Programming:

When two or more people code together e.g: Github.

* Planning Game:

Release Planning:

Requirements are gathered and commitments are done regarding project.

wh req should be implemented first and how quickly you have to deliver the product

Iteration Planning:

The developers are involved to plan the activities and tasks for iteration.

* Test Driven Development:

Unit testing is done - every smaller chunk is tested at every phase

* Whole team:

Customers and end users are also involved in it.

02. Continuous Process:

* Continuous Integration:

Smaller chunks are integrated.

Adv * Less Documentation, saves time
* The developers create extremely simple code which can be improved at any time.

Dis Focus on code not on design → market strategy.

28

* Design Improvement:
The design is further improved at every release.
* Small releases:
Small releases for frequent feedback, tracking and reduce chances of errors.

03. Shared Understanding:
* Coding Standards:
We follow certain formats so that the chunks are not dispersed.
* Collection Code Ownership:
All people working on the project are owners. Everyone is allowed to make changes in it.
* Simple design:
System Design should be kept simple. It is easier to understand and refactoring and collective ownership is made possible.
* System metaphor:
The system's structure is kept easier to communicate and elaborate.

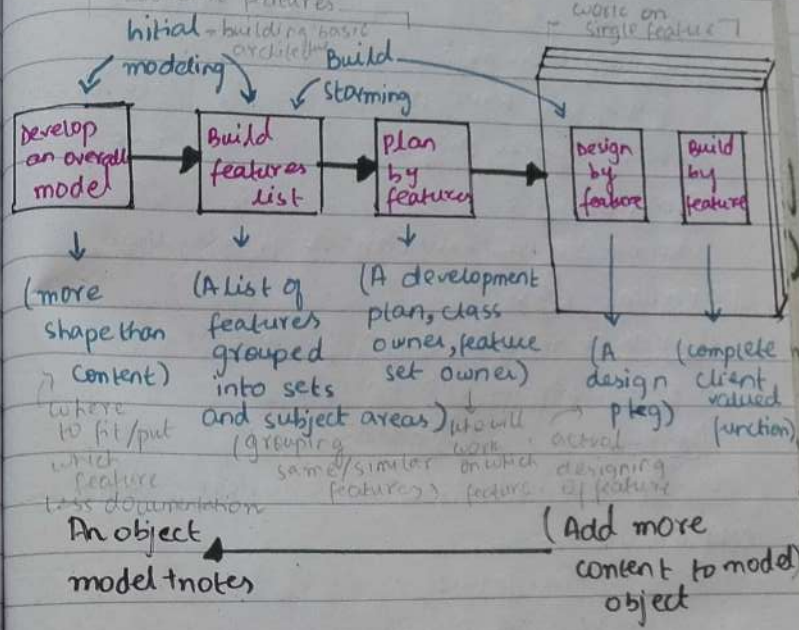
04. Programmer welfare:
* Sustainable pace:
XP emphasizes on the limited no. of hours per week for every members based on their sustainability.

CODEISE

5/6/17

29

* Feature Driven Development: (It means we are totally dependant on features)



realistic

* Client centric, architecture centric, pragmatic software process.
* first 3 steps are done in Iteration 0, the good being to identify scope of effort, initial architecture and initial high level plan.
* 6 primary roles on FDD project:
Project manager, Chief Architect, Development manager, Chief programmer, Class owner, domain expert.

-> From Client requirements we extract features.

-> Feature set owner:

For eg we have made set of features of database owner only they can configure the DB

-> Class Owner:

Same class Owner can configue that particular class

-> Iteration means no. of features we are working or in one go.

-> Development Manager:

Which work will be done by whom, keep an eye on teams.

-> Chief Programmer:

He is feature set owner

-> Domain expert:

One who verifies the features

-> Feature is the functionality. It must contain action, result, object. Like on clicking button what will happen that is feature

10 July 2017 Lean SW development:

Translation of lean manufacturing and lean IT principles and practices.

There should be lean development in software. The products should be made without any wastage. Three types of wastages were identified:

Muda: Anything not adding value to the customer. Muda is the most common wastage.

Muri: Waste created through overburden.

Mura: Waste created through uneveness.

Muda:

e.g: . Irrelevant documentation.

. Excessive Built-in apps in mobiles which are not adding any value to the customer.

Muri:

e.g: . Time loss

. Cost will increase

. Quality / Efficiency of the product will reduce.

and the loss due to these factors will be a wastage.

Mura:

e.g: . If the project work is not divided equally.

. If the time is not properly managed and if the resources are not properly managed, then there will be uneveness.

Perfection in the product comes through the reduction of non-value added activity but also through the smoothing of flow and elimination of overburdening.

Lean Principles:

01. Eliminates waste.

Unnecessary code, features and functionality are eliminated.

02. Amplify Learning.

Create solutions to unique customer problems means new work should be done everytime.

03. Decide as Later as possible.

As customer's needs are not always clear or understood.

04. Deliver as fast as possible

Rapid delivery means less time for customer to change their mind.

05. Empower the team

Let the working team design their own working procedures.

06. Build Integrity → Perceived Integrity (just to show to customer)
→ Conceptual Integrity (actual work)

* Perceived Integrity:

short iterations should be used & feedback should be acquired.

* Conceptual Integrity:

Understand the problem and solve it at the same time

07. See the whole.

(Think big act small, fail fast learn quickly)

You must have a complete picture of your project.

Disadv
- Involves meetings at every stage.
- Sometimes require a new idea/strategy.

12 July 2017 34

Scrum:

Scrum is the most used technique in Agile methodology. Scrum is used in the following cases:

- Aggressive deadlines: when you have to deliver the project in exact time, so the scrum will be the best choice
- Complex requirement: when the project is very large and all the req are not known
- Design of uniqueness: If you are working on a new innovative idea w/c doesn't exit already

For a new project, you need a new ideology or strategy, so, the scrum will be the best choice.

Project moves forward in series of iterations called Sprints (2-4 weeks)

Scrum includes:

- Scrum Team:

Scrum team is comprised of 5-9 people approximately.

- Product Owner:

are the stakeholders who create a prioritized wish list called product backlog.

- Scrum Master:

Scrum master leads the project and keeps the team focussed on its goals.

Scrum is the best choice for people-centric approach and can be used for the new ideas proposed that's it is frequently used technique of Agile Methodology.

35

Scrum Events:

- Product Backlog:

The prioritized wish list was created by the product owner in document form

- Sprint Planning meeting: SPM is conducted in w/c The team decides that how the iterations will be done in one sprint.
- Daily Scrum:

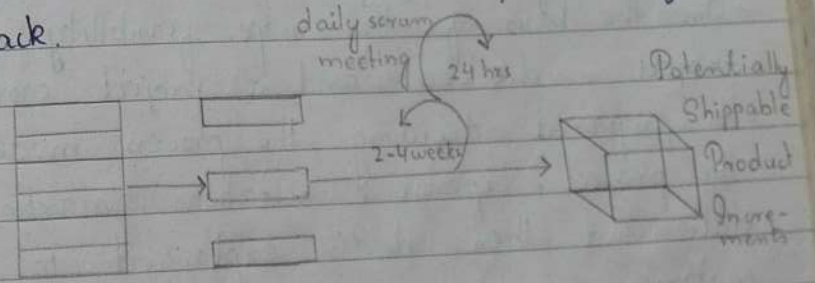
Each day the team's progress is assessed by the Scrum Master called daily scrum

- Scrum Review meeting:

The whole work is reviewed once.

- Sprint Retrospective:

The work is presented to the product owner and ask for the feedback.



Adv:
- Working software is delivered in 2-4 weeks.
- The product is reviewed at each phase.
- Provide flexibility.

17 July 2017

Software Requirement Engineering:

Functions, condition / limitation, behaviour, features, user's expectation to gather all these requirements. Anything that customer wants you to do.

SRE Process:

The procedure from requirement gathering to a documentation phase is called SRE Process.

Q: When does the project open??

Project Opening:

→ The very first meeting with the customer is scope of the project.

→ On the basis of this scope, feasibility of the project is checked and a project contract is prepared deciding the process model for the project, expense, limitations, starting date and then it is explained to the customer.

→ Once the project contract is signed, the project is now open.

→ Then starts requirement gathering. If the customer tells the requirements before the contract is signed, the idea might get leak.

Q: When will the project close??

Project Closing:

→ When all the requirements are converted into working functionality.

→ When the project is deployed and the customer gives the feedback.

→ When the maintenance period is over.

Tasks:

Feasibility study → Inception

Requirement Elicitation → Analysis
→ Elaboration
→ Negotiation.

Specification → Specification

Requirement Validation → Requirement Management.

Elicitation → Gathering
→ Org
→ Neg

whether the system could be made or not. The validity of the req is checked. Its budgetory constraints are checked.

whether the project is ensures that technically & logically feasible or not

38

① Feasibility study: How to gather the requirements. We find out the already existing features. This is also called Inception.

19 July 2017

② Requirement Elicitation:

Details are gathered from client. To gather requirement we can use following methods:

strong medium for req gathering

(i) Interviews:

↳ Oral Interviews ↳ Written
↳ One to One ↳ Group.
↳ Structured / Closed Interview.

All questions are list down before interview.

↳ Non-Structured / Open Interview:

The questions are not list down before interview.

Type of interview depends upon nature of project and the level / designation of client.

(ii) Questionaires:

Requirement gathering through

21 July 2017

39

questions where your presence is not necessary. Questions are just sent to the targeted people.

(iii) Survey form: is used

→ When the product is general.
e.g: Game
→ Surveys depends upon the no. of sample (response)

(iv) Task Analysis:

What should be the simplest way to do the task. Task analysis is done when customers usually have no knowledge of the product.

(v) Observation:

Observation phase is done if the system already exists.

(vi) Brainstorming:

Stakeholders, customers and the designers give their suggestion regarding project. It is the best approach when you have a new and

creative idea.

(vii) Prototyping:
An early approximation of the product. It is used when all the requirements are not known.

⑤ Requirement Organizing:
The requirements are filtered and grouped in different classes.

(i) Negotiation:
Conflicting requirements are negotiated (discussed)

(ii) Specification:
An SRS document is prepared in it and requirements are validated.

④ Requirement Management:
Management phase is done if you have to reconduct the SRE again and again.

Type
- User Req
- System Req
  - Func
  - Non-func
  - Domain

24 July 2017.

Requirement Types: are made so as to easily assemble the requirements.
- Functional
- Non-Functional
- Implicit
- Explicit

} major system requirements

SRS document:
in w/c The requirements are written by following certain standard like priority wise or the similar requirements are grouped together.

01. Functional Type:
which highlights the functionality of your projects They show behaviour of your projekt.
e.g.
→ Login and Logout pannel
→ Registration option in forms.
→ Code of function behind button click

They are easy to verify but difficult to change.

02. Non-functional Type:

These are extra requirements. Like. To provide certain formats and standards for data security and redundancy. They don't show behaviour of project.

e.g: → Reliability → Portability → Scaleability
→ Interface designing
→ Different SRE security techniques.
→ Performance - throughput, response time, utilization.

They are difficult to verify but easy to change.

03. Implicit:

Requirements which are usually taken from customer. They are not known by the developers.

e.g:

- Storage of the product (Implicit + Non-functional)
- Tables in the DB
- Interface designing
- Theme . Color . Features

04. Implicit:

Requirements which developers gather for themselves according to the user requirements. They are known by the developers.

e.g:

- Responsiveness of product.
- Language used in the project.
- Data Security
- Like, user expect that their password won't be stored in plain text. This req → don't have to be written down by client.

Domain:-

Domain knowledge is the knowledge about the domain you are creating a solution for. For e.g (If you are creating an app for a bank then knowledge about banking (accounting, banking procedures) is the domain knowledge

26 July 2017

44

Functional Requirement	Non-functional Requirement
• Describe what software should do.	• Describe how sw will do so.
• Example: System must send e-mail when condition met behaviour of system.	• Example Email should be send with latency of ~~on~~ greater than 2 hours.
• Depends on types of software, expected users, type of system where sw will use.	• Performance characteristic of system. • More Critical. • Product Organization External

• Typical functional req includes
- Busines rules
- Authentication
- External interface

• " non-func.
- Security
- Maintainability
- Useability
- Scaleability

From SE

45

*

Waterfall model is called traditional model because it follows classic approach in w/c activities of one phase must be completed before going to the other. It is the simplest of all models.

* Diff b/w Agile & Spiral.

Agile:
- More risk or sustainability & maintenance.
- Min rules, documentation easily employed
- Easy to manage
- Suitable for small projects
- Depends heavily on customer interaction.

Spiral:
- Better risk mangement
- Large no of intermediate stages requires excessive documentation
- Management is more complex
- Not suitable for small or low risk projects
- Does not depend heavily on customer interaction.

* Diff b/w incremental & iterative:

One app is a software development where the model is designed, implemented and tested incrementally (a little more is added each time) until the product is finished.

Iterative app is a cyclic approach or process of design, dev, testing and refining a product. Based on the results of testing the most recent iteration of a design, changes and refinements are made.

* What is Manifesto of Agile approach?

Agile Manifesto also called the Manifesto for Agile Software Development is a formal proclamation of 4 key values and 12 principles to guide an iterative & people-centric approach to software development

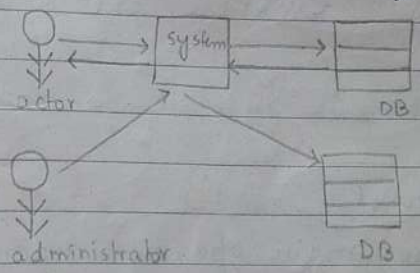
3 Aug 2017

Use Case:

Use case is used to define the interaction between the actors and system.

Importance:

→ Workflow can be identified / defined.
→ Deficiency in the requirements and functionality can be identified.
→ Gives an overall picture of the system.
→ User's right to access the features of system can be identified.



→ Use Case can be made for any process model and at any phase. Actually use in requirement phase.

• Use case is the type of UML

→ Defects in the system can be detected.
→ Help to extract more req

Elements of usecase:

- Name should match with the system.
- Brief Des Definition of system.
- Actors who gives an input (user)
- Pre Condition conditions necessary to enter in user
  → To withdraw cash, you must have card
- Basic flow You give an input and the system respond. To follow proper system.
- Alternate flow If you enter wreng password, the user interaction from the system eliminates
- Exceptional flow. Exceptional case
- Post condition

↓ when the system changes its mood

Trigger:

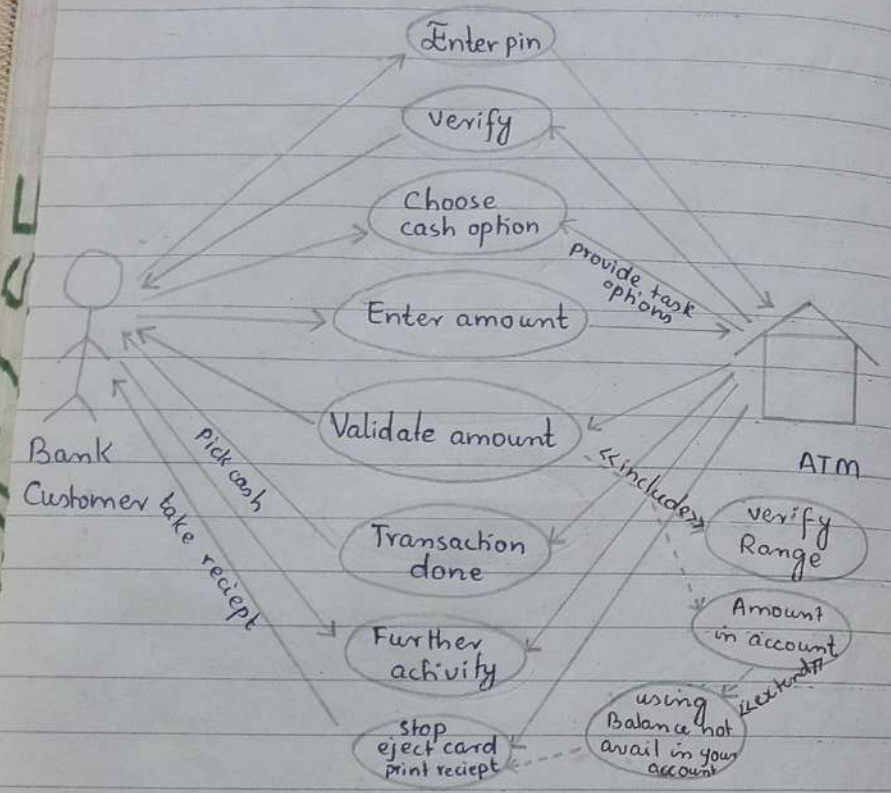
Change of event

↳ Something whose main operation depends upon particular event

↳ Those events on w/c software working depend.

↳ Valid conditions to activate systems working

4 Aug 2017 Example of Use-Case: 50



Name: ATM money cash withdrawal

Description: User enters pin number which will be verified by ATM and user is provided with different options. User choose options and enter amount. ATM again validate amount and return cash to user with option to continue or stop transaction.

Trigger: change in operation/event (Trigger is basis of the software system The system remain inactive until the triggers are activated) eg. valid admit card (pre-condition), 5:1 time to start paper (trigger) e

Actors: Bank customer, ATM

Pre-condition: Must insert ATM card

Post-condition: Cash with draw

Normal flow:

1. User enter pin no.
2. ATM verify pin.
3. ATM provide cash options.
4. User select appropriate options or enter amount
5. ATM machine validate the amount
6. Return the cash to the customer
7. If you want to choose any option from the menu again or if you again want to withdraw cash. Select option.
8. Tasks are done. Now eject card and get the reciept.

Alternate-flow: Step 5 A1:

1. Machine checks for allowed range.
2. Machine checks whether the required amount is in account or not.
3. If required amount is not in account machine will show message and move to step 8

<<includ>> and << extend >>
are used for the alternate flow.

7/Aug/2017. How the logic of the system will be built?

Analysis Modeling:

- Provides first technical representation of system. The requirements are logically defined here.

Objectives:

- 3 primary objectives:
  - → to describe what customer requires.
  - → to establish basis for creation of sw design.
  - → to define set of requirements that can be validate once sw is built.
    - Validation: (How to do) to check functional aspect
    - Verification: (What to do) to check non-functional aspect
- Differenciates b/w essential info v/s simple info.
- provides tool other than narrative text to describe sw logic and policy.
- Shows 3 aspects of sw.
  - → Data Modeling:

(Data is modeled ↑means that how the work will be done and which work will be done on which data.)

Data modeling means that how the data items relate to each other and how they proceed and stored inside the system.

- → Functional: is a structured representations of func (activities, actions, processes, operations) within the modeled system or subject area. Functionalities of the system are defined.
- → Behavioral:

How the functionalities will behave or respond. Shows the interaction b/w obj to prod. use some particular system behaviour i.e is specified as use-case.

Analysis Rules for thumb:

- The model should focus requirement that are visible within problem or business domain.
- Each element of analysis model should add overall understanding of Requirement and provide insight into info domain, function and behaviour of system.
- Model should delay consideration of infrastructure and other non-functional model until design phase.
- Should provide value to all stakeholders.
- Should be kept as simple as can be.
- The model should minimize coupling throughout system.

Area of work

Domain Analysis:

Interfaces are not made.

The identification, analysis & specification of common, reuseable capabilities within specific app domain in terms of common objects, classes, subassemblies and framework.

- Source of domain knowledge:
  - Technical literature. (classes, libraries)
  - Existing app. (If any similar app exist, you can get the process flows of app) (when you are making any generic product)
  - Customer survey and expert advice.
  - Current / future req.

- Outcome of domain analysis.
  - Class taxonomies. What class, objects and liabraries should be made.
  - Reuses standards.
  - F/n and behavioural model.
  - Domain language.

Analysis Modeling Approaches:

Structured Analysis: → can be done for C lang.

- Consider data and processes that transform data as separate entities.
- Data is modeled in terms of attributes & relationships (no operation)

Object Oriented Analysis: → works on operation. Operations are also discussed.

- Focuses of def of classes and manner in which they collaborate with one another to fulfil requirements.

Elements of Analysis Model:

Object-Oriented Analysis.

Senario based
Usecase text
Usecase diag
Activity diag
Swin Lane diag

Class Based
Class diag
Analysis Pkg
CRC Model
Collaborative diag

Structured Analysis

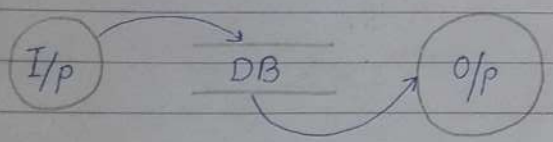
Flow Oriented
Data structure diag
Data flow
Control flow
Processing Narrative

Behaviral
State diag
Sequena diag

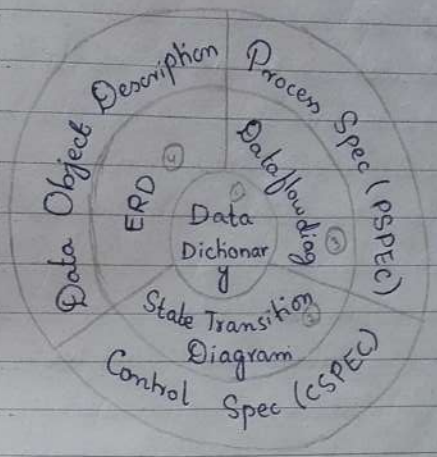
Flow Oriented.

Provides indication of how data objects are transformed by set of processing function.

- Scenario based representation system from user point of view. (Use-case diagram)
- Class based. Define objects, attributes and relationships. (Class diagram)
- Behavioral depicts states of classes and impacts of events on these states.



16 Aug 2017.



Data Modeling:

- Data Object
- Data Attribute
- Relationship
- Cardinality
- Modality

- State Transition Diagram is behavioural which is part of OO diagram.
- This all info is obtained through structured analysis.
- Data Dictionary is created at the start of analysis.

(1) ER Diagrams: entities are classified and then relationships are defined. (To produce a conceptual data model of an info system)

• Data Object Description:
Contain Detailed info about object.

③ Data Flow Diagram: from external entity to system
show data flow and it requires Process Specification that w/c process is followed at the back end.

② State Transition Diagram:
How your system behaves when the state changes. It is part of behavioural model. (OO approach)
which operation will be performed on that particular event. It requires Control specification: All those events / thing which will bring change in the state are defined here.

Data Modeling requires above 5 elements

* Data Objects:
are the entities. It can be a real world entity.

* Data Attributes:
are the properties or characteristics of object
shows sche/aspects of data objects

* Relationship:
links b/w the entities.

* Cardinality:
No of occurances in a relationship

* Modality:
Whether the existing of data object is mandatory or optional.
1 is used for mandatory
0 is used for optional.

① Data Dictionary: a set of info describing the contents, format and structure of a database and the relationship b/w its elements

SW design:

An iterative process transforming requirement to a blueprint for constructing Software.

- Goals of design process.
- Design principles.

Goals of design process:

- The design must implement all of explicit requirement contained in analysis model and it must accomodate all of implicit requirement desired by customer.
- The design must be readable, understandable for those who generate code and for those who test and support software.
- The design should address data, functional and behavioural domains from implementation perspective.

Design Principles:

- The design process should not suffer from tunnel vision.

- The design should be traceable to analysis model.
- The design should not reinvent the wheel.
- The design should minimize the intellectual distance between software and problem as it exist in real world.
- The design should exhibit uniformity and integration.
- The design should be structured to accomodate change.
- The design should be structured to degrade gently even when ~~ap~~ aberrent (false) data, events or operating conditions encountered.
- Design (↗ High abstraction level) is not coding. Coding is not design. (↘ low abstraction level)
- The design should be assessed for quality (↗ LOC (Line Of Codes) are used. Low LOCs mean better quality) as it is been created not after the fact.
- The design should be reviewed to minimize conceptual (semantic) errors. (↓ errors before coding)

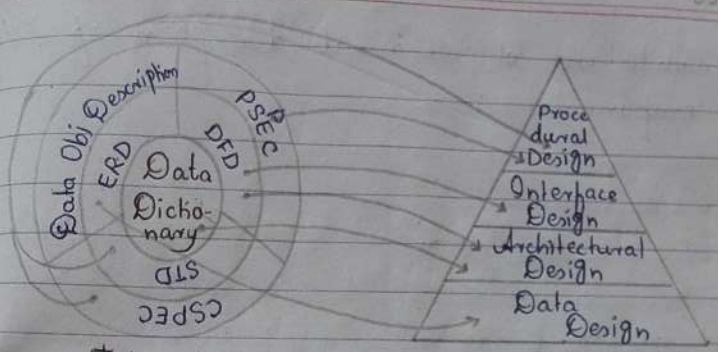
- SW design o/p is validated
- Analysis o/p is verified.

GOALS:

- Explicit requirements:
  User's requirement
  Implicit requirement:
  Req w/c user expects you to fulfil.
- Writing algorithm is a part of design.
- Design is the combination of these 3 domains.

PRINCIPLES:

1. Tunnel vision:
   Single path for design.
2. Design should be the reflection of analysis model.
3. Do not build anything from scratch.



Translating analysis model to design:

Components of Design Model:

- Data design transforms info domain model to data structure requirement to implement sw.

* Architectural Design:
  Define relationship among major structural elements of software.

* Interface Design:
  Describe how software communicates with systems and human that interact with it.

* Procedural Design:
  Transforms structural elements of

architecture to procedural description of SW components.

28 Aug 2017

Design Concepts:

The fundamentals of software design concepts are as follows:

01. Abstraction:

The concept of abstraction is used to hide unnecessary details from the users. Usually a high-level of abstraction is used while stating a solution in large terms.

- **Procedural Abstraction:**

A sequence of instruction that contain a specific and limited function refers in a procedural abstraction.

e.g login pannel steps are hidden.

- **Data Abstraction:**

A collection of data that describes a data object is a data abstraction.

- **Control Abstraction:**

If using control for any task, it should be hidden

02. Refinement:

Refinement is a top-down design approach. It is a process of elaboration in which the procedural details are refined. There is a low-level of abstraction.

The functional statements are decomposed and refined.

03. Modularity:

A system is divided into addressable components. They are called as "modules". Modularity permits a program to manage easily.

- Modular decomposibility:

The major problem is divided into sub-problems and a separate design is made for each sub-problem

e.g: Pay Roll System → Salary generation
Pay deduction

- Modular Composibility:

The already built component (useable component) is combined with new work and then the working is ~~divided~~ done on task.

- Understandibility:

The work is divided in order to maintain ease.

- Continuity:

Division on the basis of changes. The frequently changing components are modeled together

- Protection:

The most faulty or errorful section is modeled separately.

30 Aug 2017

04. Software Architecture:

The complete structure of the sw is known as software architecture showing how the module will interact each other.

- Structured Model:

Workflow is not defined just structured is given.

- Framework Model:

It states that how the activities will be defined and in which manner. The aim of sw design is to obtain an architectural framework of a system.

- Dynamic Model:

Also called behavioural model states that how the individual components will behave

- Process Model:

Dividing the work in steps and stages.

• Functional Model:

Features and functionality of the components is defined.

05. Control Heirarchy:
(Program Structure)

Represent the organization of program components and implies a control heirarchy. It does not represent the procedural aspects of the software (event sequence).

e.g: Classes, Methods, relationships, workflow is defined here.

06. Structural Partitioning:

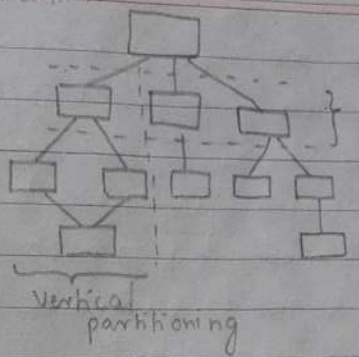
The heirarchal program structure can be partitioned into:

• Horizontal Partitioning:

Horizontal partitioning defines 3 partition
• input • data transformation • Output

• Vertical Partitioning:

Vertical partitioning (factoring) distributes control in a top-down manner.



Horizontal approach is better if we have used same data structures for all pannels

07 Data Structures:

Shows the logical relationship among individual data elements.

e.g: Arrays, Stacks, Linked Lists etc.

08. Software procedure:

shows the precise specification of processing like event sequence, decision points, repeatitive operations, data structures

e.g: Binary search algorithm for searching praedure

09. Information Hiding:

Modules must be specified and designed so that the information like algorithm and data

presented in a module is not accesible for other modules not requiring that information.

Scope of Scenario: (10-12 lines)

- Why do you need that project?
- Who is the targetted audience?
- What benefits do you expect
  - For advertisement - For jobs.

30th Aug 2017

SW Testing:

Process of executing program with intention of finding errors.

Purposes:

if fulfuling req → reasonable cost

01. To demonstrate quality or proper behaviour.
02. To detect and fair solve problems.

Major Activities:

01. Test planning and preparation.
02. Test extension 03. Analysis & follow up

Objectives:

Direct:

- To identify and reveal as many errors as possible.
- To bring tested SW after correction of errors and retesting to an acceptable level of quality
- To perform required test efficiently & effectively within budgetary and sheduling limitation

Indirect:

- To compile a record of SW errors for use in error prevention.

02. Test execution:

Diff techniques are used:

Test Case: How the system will behave on particular i/p.

Test Stress: Max o/p the system can give.
· e.g inserting more records than declared size.

Analysis: Recheck the code. whether the error is valid or not.

Follow Up: Whether the error correction is right or wrong.

6 September 2017.

01. Black Box (functional testing):

Black box testing is also called Behavioral testing. It is a software testing method which is used to test the software without knowing the internal structure of code or program.

Example:

A tester, without knowledge of the internal structure of a website, tests the web pages by using a browser; providing i/ps (clicks) and • verifying the o/ps against the expected outcome.

• It focuses on final requirement.
• This method is an attempt to find errors in following categories.
1. Incorrect or missing function.
2. Interface errors.
3. Errors in Data Structures.
4. Behaviour or performance error.
5. Initialization and termination error.

Advantages:

• Tests are done from users point of view.
• Testers do not need to know any programming language as he has no concern with

internal structure.

Disadvantage:

- Test cases are difficult to design. as it is difficult to identify all possible i/ps in limited testing time.

02. White Box Testing:

Also known as clear box testing, glass box testing, transparent box testing and Structural testing. It is software testing technique in which step-by-step execution is done so that it reveals errors in "hidden" code. It is concerned with the internal working of system. ~~testing~~.

Disadvantages:

- Expensive and time consume.
- If code is large, there is possibility of missing any line or some lines accidentally.
- Vast resource utilization

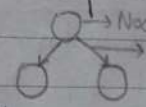
It

- is concerned with internal mechanism.
- examine internal calculation path.

03. Basis Path testing:

This testing method analyzes the control flow graph of a program to find a set of linearly independent paths of execution

Steps to find/Calculate the independent paths:



1. Draw flow graph → bubbles are used
2. Calculate cyclometric complexity
3. Identify paths. 3 paths
4. Prepare Test Cases.

Cyclometric Complexity:

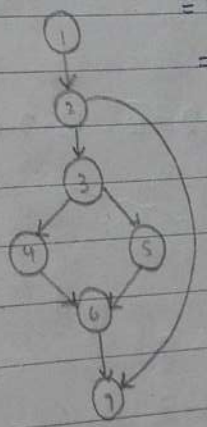
Determines the complexity level of program. It helps in identifying alternate paths from beginning till end.

$V(G) = Edges - Nodes + 2$

$= 8 - 7 + 2$

$= 3$ paths

1, 2, 8




```
int a;
for(i=10)
{
if( )
else
{ }
}
exit:
}
```

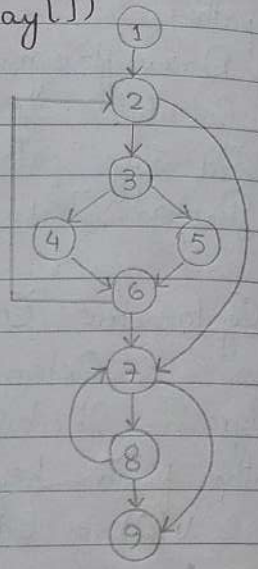
For test cases we need,
Path, Input data, Expected o/p.

1, 2, 8 i = 11 terminate

6th Sep 2017

Example: not mandatory

```
del (int value, int size, int array[])
{ int i;                          } initialization
  loc = size+1;                   1
  for i=1 to size                 2
  if (array[i] == value)          3
    loc = i;                      4
    end if;                       6
  end for;
  for i = loc to size             7
  array[i] = array[i+1];          8
  end for;
  size--;                         9
}
```



V(G) = E - N + 2 ⇒ 13 - 9 + 2 ⇒ 6

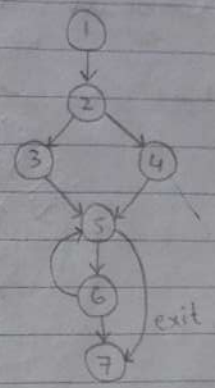
Path:

1: 1-2-7-8-9 ✓
2: 1-2-7-9 ✓
3: 1-2-3-4-6-7-8-9 ✓
4: 1-2-3-5-6-7-8-9 ✓
5: 1-2-3-4-6-7-9
6: 1-2-3-5-6-7-9

Example: 02

Q: Void f1 (floaty, float a, int n)

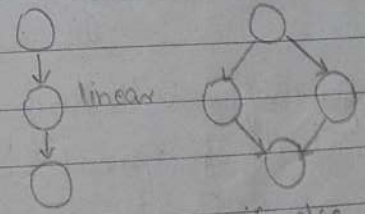
```
{ float x = y * n;          1
  if (x > 0.01)             2
  z = tan(x);               3
  else
  z = Cos(n);               4
for (i=0; i<max; ++i)       5
{ a[i] = a[i] * 2; }        6
  return [i];               7
}
```

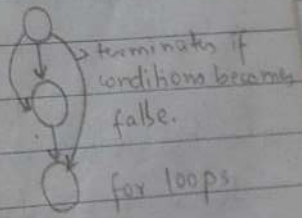


V(G) = E - N + 2 ⇒ 9 - 7 + 2 ⇒ 4

Path:

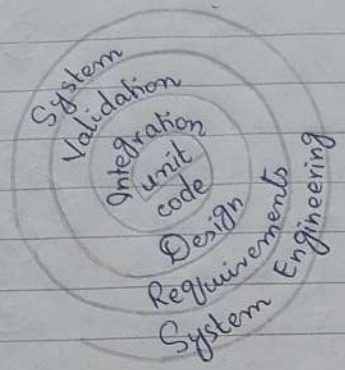
1: 1-2-3-5-6-7
2: 1-2-4-5-6-7
3: 1-2-3-5-7
4: 1-2-4-5-7





8th September 2017

Overall testing Strategy:



This spiral shows the software engineering process like the first spiral i.e System Engineering defines the role of the software and leads to requirement analysis where domain, function, behavior, performance, constraints are established. Then design of the requirements is prepared and finally converted into coding.

A strategy for software testing can also be viewed in the context of spiral. The code is divided into small units. These units can be designed in parallel levels and then you can apply any testing

strategy (Black Box or White Box). This is called "Unit Testing" in which internal mechanism is tested in small units. The units are then gradually combined and tested simultaneously. For dependent units, different supporting tools are used to combine them.

For Integration, we have following testing techniques:

01. Integration Testing:

In this, The units are combined and tested at the same time to uncover errors associated with interface.

- Top-down approach:

Main control module is tested first then moving downward to check lower modules through control heirarchy.

- Bottom-up approach:

Lower modules are tested first then moving upward to check main control module through control heirarchy.

• Regression Testing:

Regression Testing is done to ensure that changes have not affected the system and the system is working properly.

02. Validation Testing:

It is done to check whether the system is perfectly working or not as per user requirements.

• Alpha testing:

The alpha test is conducted at the developers site by a customer. It is conducted in a controlled environment.

• Beta Testing:

It is conducted at one or more customer sites by the end-user of the software. Reviews are taken but it is a bit complex task as the reviews collection might take long time. It is done in open environment.

03. System Testing:

System testing is done to verify that all system elements are properly integrated and working.

• Security Testing:

It is conducted to verify that protection mechanism built into a system will protect it from improper penetration.

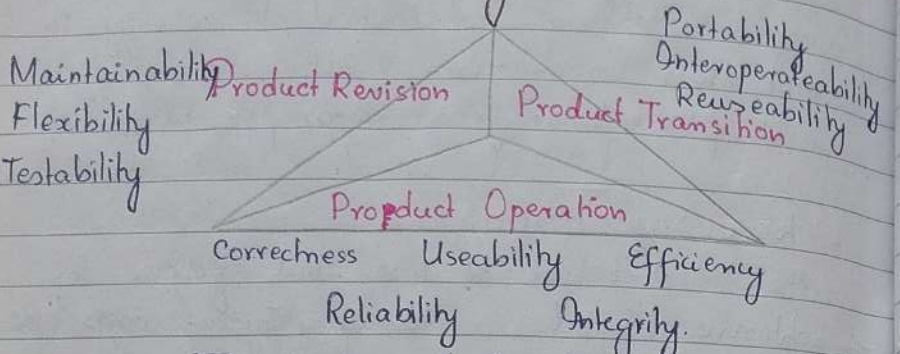
• Recovery Testing:

It is a test that forces the software to fail in a variety of ways & verifies that recovery is properly performed.

• Stress Testing:

Maximum inputs are given to check the flexibility or stress level of system. If we enter the data more than declared size, then performance, response time is checked through stress testing.

McCalls Quality Factors:



The factors that affect the quality of the software focus on 3 important aspects of a software product:

1. Product Operation:

The operational characteristics of system including its functionality and behaviour.

2. Product Transition Revision:

The ability of a software product to undergo change.

3. Product Revision Transition:

The adaptability of a software product to new environments.

These 3 aspects include the following description:-

- Correctness:

The extent to which a program can be satisfies

expected its specifications and fulfils the customers mission objectives.

- Reliability:

The extent to which a program can be expected to perform its intended function with required precision.

- Efficiency:

The amount of resources & code used by a program to perform function determines efficiency.

- Integrity:

How The extent to which access to software is controlled.

- Useability:

Efforts required to learn, operate, prepare i/p and interpret o/p of a program.

- Flexibility:

Efforts required to modify an operational program.

- Testability:

Effort required to test a program to ensure that it performs its intended function

- Maintainability:
  Efforts required to fix an error.
- Portability:
  Efforts required to transfer program from one environment to another.
- Reuseability:
  Extent to w/c a program can be re-used in other applications.
- Interoperability:
  Efforts required to couple one system to another.

11 September 2017.

RISK MANAGEMENT:

RISK: Uncertainity / Expectation reach to danger / Probability of occuring unwanted events.

Risk Management:

The risk management is the process of identifying, analyzing, assessing and preparing a contigency plan to minimize the risk. The risk can't be fully eliminated but can be minimized. The risks which have high impact factor are handled first. For risk management we need following factors:

* Risk * Impact * Probability / Frequency

Reactive Strategy:

No pre-planning is done for Risk Management. The software team does nothing about until something goes wrong. Then, team flies into action in an attempt to correct the problem rapidly. This is often called "Fire-Fighting Mode". Sometimes situation becomes out of control.

Proactive Strategy:

Pre-planning is done for risk-management. The probability of risk occuring and their impact is identified and assessed. The software team establish contigency plan to avoid risk. that will enable it to respond in a controlled and effective manner.

Type of Risks:

- **Project Risk:**

It threaten the project plan. It is concerned with scheduling, budgetory, personnel, resource, req gathering problems etc.

- **Technical Risk:**

Technical risks are concerned with performance, quality and timeline of the software to be produced.

- **Business Risk:**

is concerned with the viability of the software to be built. It is categorized as:

- **Market Risk:**

Building a product that no one really wants.

- **Strategic Risk:**

Building a product that no longer fits into the overall business strategy for the company.

- **Sales Risk:**

Building a product with not enpected sales.

- **Management Risk:**

Not proper team handling associated with software project. Not proper schedule handling.

- **Budget Risk:**

is concerned with loosing budgetory

- **Known Risk:**

Mandatory Risks, which have high probability of occuring, which are uncovered after careful evaluation of project.

- **Predictable Risk:**

are explored from past experience of SW project.

• Unpredictable Risk.

which are difficult to identify in advance.