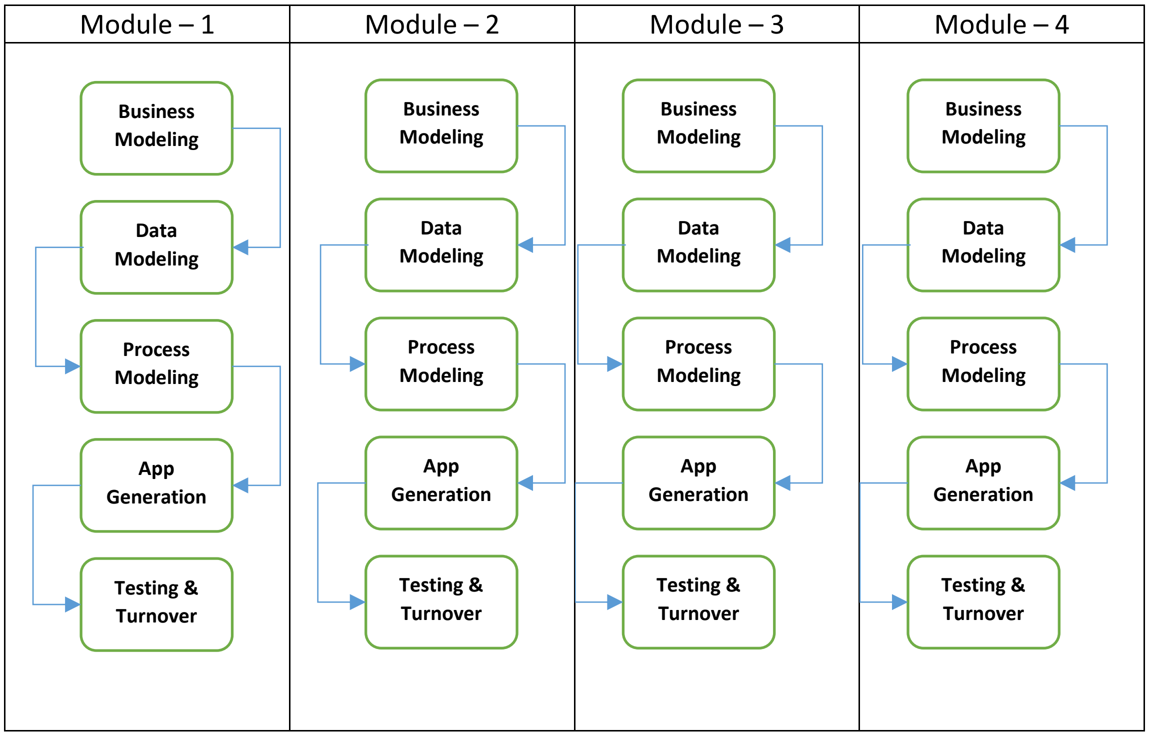
NOTES WEEK FOUR
21-Mar-2022 TO 25-MARCH-2022

A. RAD Rapid Application Development

a. RAD is a linear sequential software development process model that emphasizes a concise development cycle using an existing element based construction approach.
b. System is divided into modules. Each module has its own lifeline.
c. Reuse of components developed earlier are used as starting line or are used for prototyping.
d. Schedules are very strict and no-delay policy is part of the plan.
e. Less horizontal communication among teams.



PHASES of RAD:

1. BUSINESS MODELING

The information flow among business functions is defined by answering questions like what data drives the business process, what data is generated, who generates it, where does the information go, who process it and so on.

2. Data Modelling:

   The data collected from business modeling is refined into a set of data objects (entities) that are needed to support the business. The attributes (characteristics of each entity) are identified, and the relation between these data objects (entities) is defined.
3. Process Modelling:

   The information object defined in the data modeling phase are used to create a flow like situation. The start of data and later transformations are linked in a linklist like format. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object. (CRUD-Principal)
4. Application Generation:

   Automated tools are used to facilitate construction of the software. Tools like 4th GL techniques.
5. Testing & Turnover:

   Many of the programming components have already been tested since RAD emphasis reuse. This reduces the overall testing time. But the new part must be tested, and all interfaces must be fully exercised.

WHEN TO USE:

i) When the project can be divided into modules.
ii) When modules realization time is a short span (2-3 months).
iii) When the requirements are well-known.
iv) When the technical risk is limited.
v) It should be used only if the budget allows the use of automatic code generating tools.

ADVANTAGES:

a. Model is flexible
b. Change adoption is easy
c. Delivers higher quality to customer
d. Short development time
e. Reusability of components

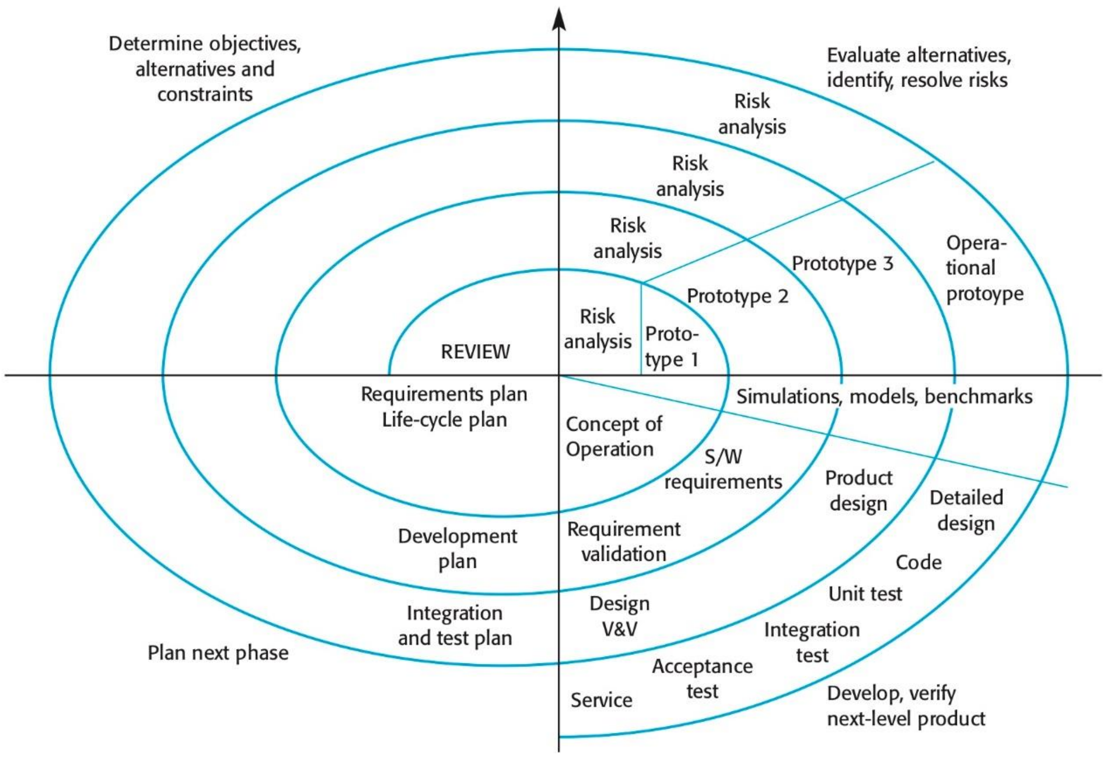
DISADVANTAGES:

a. Requires a highly skilled design team for modules. (Adaptive design, Reusability, Reverse engineering, Pattern based design etc.)
b. Not all projects can be done using RAD. (can NOT be used repeatedly for successive projects of different/heterogeneous nature)
c. Not suitable for small size projects
d. If technical risk is involved the model fails
e. Requires an intelligent and well informed used for feedback and constant interaction.

The SPIRAL MODEL (BOEHM'S SPIRAL MODEL)

The spiral model, initially proposed by Boehm, is an **evolutionary software process model** that **couples the iterative feature of prototyping** with the **controlled and systematic aspects of the linear sequential model**.

a. Software RELEASE is done in INCREMENTS
b. Each release produces an outcome (paper document, paper model, prototype, semi-functional module etc.)
c. With each iteration of a development cycle product become more and more strong and fuller.



Understanding the Spiral:

Each cycle in the spiral is divided into four segments:

Objective setting:

Each cycle in the spiral starts with the identification of purpose for that cycle, the various alternatives that are possible for achieving the targets, and the constraints that exists.

Risk Assessment and reduction:

The next phase in the cycle is to calculate these various alternatives based on the goals and constraints. The focus of evaluation in this stage is located on the risk perception for the project.

Development and validation:

The next phase is to develop strategies that resolve uncertainties and risks. This process may include activities such as benchmarking, simulation, and prototyping.

Planning:

Finally, the next step is planned. The project is reviewed, and a choice made whether to continue with a further period of the spiral. If it is determined to keep, plans are drawn up for the next step of the project.

The development phase depends on the remaining risks. For example, if performance or user-interface risks are treated more essential than the program development risks, the next phase may be an evolutionary development that includes developing a more detailed prototype for solving the risks.

The risk-driven feature of the spiral model allows it to accommodate any **mixture of** a **specification-oriented**, **prototype-oriented**, **simulation-oriented**, or another type of approach. An essential element of the model is that **each period of the spiral is completed by a review** that includes all the products developed during that cycle, including plans for the next cycle.

The spiral model works for development as well as enhancement projects.

WHEN TO USE:

a. WHEN the project is Large to Huge in size.
b. WHEN requirement are very complex and unclear.
c. WHEN change may occur at any time/step/phase.
d. WHEN budget for the project is huge.
e. WHEN delivery of work-products are needed at a frequent pace.

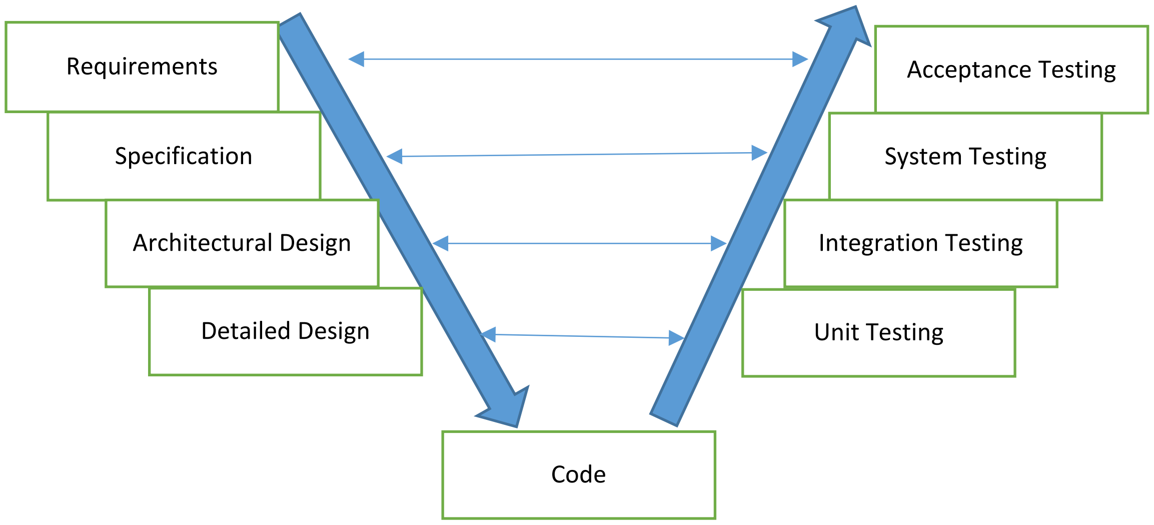
ADVANTAGES:

a. A high level of RISK analysis
b. Very useful for mission-critical and large projects

DISADVANTAGES:

a. Cost is high.
b. Risk analysis requires a special and high level of expertise.
c. Do not work for small scale projects.

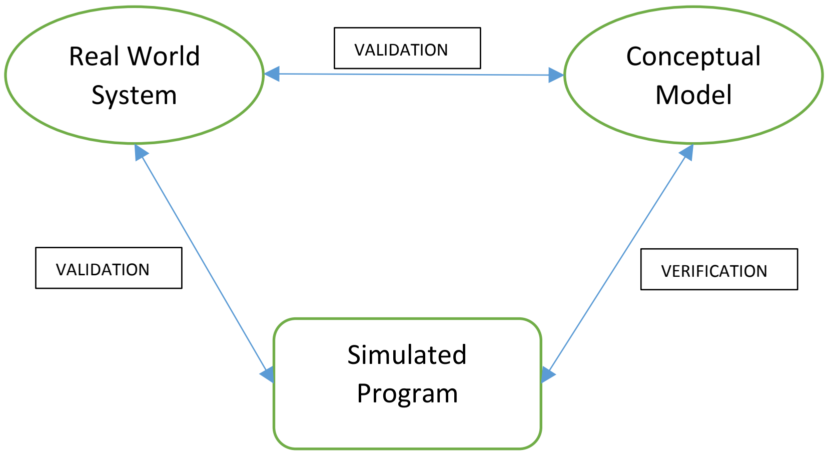
THE V-MODEL (VERIFICATION AND VALIDATION MODEL, VnV Model)



Validation asks '**Are we building the right thing**?'

whilst

verification asks '**Are we building the thing right?'**



Each phase of SDLC must complete before the next phase starts. It follows a sequential design process same as the waterfall model. Also called an **improvement over waterfall model**. Testing of the device is planned in parallel with a corresponding stage of development.

VALIDATION: Ask the customer/client/ or compare with real world system. REQUIREMENTS are checked and rechecked for this phase. (static analysis: no code involved, used to improve conceptual model, feedback) WHAT IS REQUIRED / HAVE WE UNDERSTOOD IT RIGHT / RECHECK

VERIFICATION: Use intermediate results to compare with observed values, use model to test outcomes. (dynamic analysis, code tested against validity standards, extensive testing) WHAT WE HAVE MADE IS IT RIGHT / IS WHAT WE MADE IS REQUIRED / DOES IT DO WHAT WAS THOUGHT IT WOULD

WHEN TO USE:

a. When the requirement is well defined and not ambiguous.
b. When project measures from small to medium-size and requirements are clearly defined and fixed.
c. When team has previous experience of similar projects, and sample technical resources are available.

ADVANTAGES:

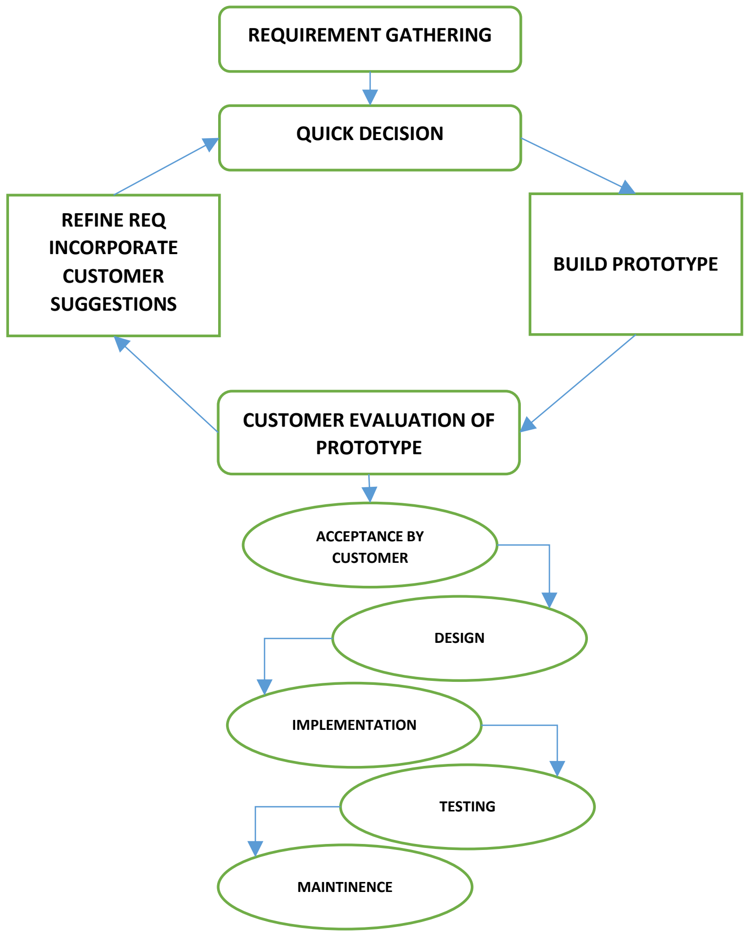
a. Easy to understand
b. Well planned and documented
c. Higher chance of success over waterfall model
d. Works well for small projects where requirements are clear

DISADVANTAGES:

a. Rigid unchanging too strong plan with little room for deviation
b. Not good for complex requirements
c. No prototyping thus feedback is absent
d. Changes are very difficult and hard to absorb

The PROTOTYPE MODEL

A prototype is a kit implementation of the system. A prototype usually turns out to be a very crude version of the actual system, possibly exhibiting limited functional capabilities, has low reliability, and inefficient performance as compared to actual software In many instances.



WHEN TO USE:

a. WHEN customer has little understanding of the requirements
b. WHEN product is very customer specific and does not have a large user base.
c. WHEN no limitation over budget and time of delivery.

ADVANTAGES:

a. Reduced risk of incorrect user requirement
b. Good where requirement are changing/uncommitted

c. Regular visible process aids management of project
d. Support early product marketing
e. Reduce Maintenance cost.
f. Early detection of errors

DISADVANTAGES:

a. An unstable/badly implemented prototype often becomes the final product.
b. Require extensive customer collaboration
c. Costs customer money
d. Needs committed customer
e. Difficult to finish if customer withdraw
f. May be too customer specific, no broad market
g. Difficult to know how long the project will last.
h. Easy to fall back into the code and fix without proper requirement analysis, design, customer evaluation, and feedback.
i. Prototyping tools are expensive.
j. Special tools & techniques are required to build a prototype.
k. It is a time-consuming process.

The BIG BANG MODEL

In BIG BANG model there is no specific process at all. Developers begin coding as soon as they get a necessary set of resources. The resources may vary in each situation. Normally the resources are team for effort, time for work, and some finance to fuel the budget. And the result may or may not be as per the customer's requirement, because in this model, even the customer requirements are not defined.

This model is ideal for small projects like **academic projects** or practical projects. One or two developers can work together on this model.

WHEN TO USE:

a. Project is small like an academic project or a practical project.
b. The size of the developer team is small
c. When requirements are not defined
d. Release date is not confirmed or given by the customer.

ADVANTAGES:

a. There is no planning required.
b. Simple Model.
c. Few resources required.
d. Easy to manage.
e. Flexible for developers

DISADVANTAGES:

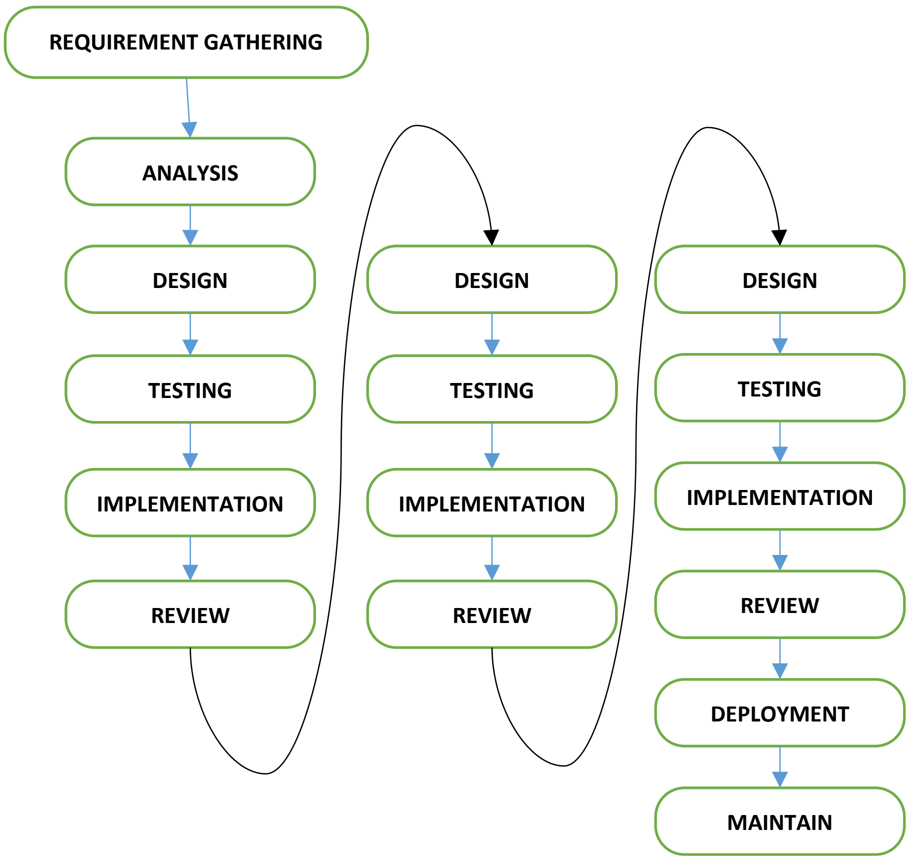
a. There are high risk and uncertainty.
b. Not acceptable for a large project.
c. If requirements are not clear that can COST expensive (YOUR GRADE‼)

THE ITERATIVE MODEL

In Iterative Model, you can start with some of the software specifications and develop the first version of the software.

After the first version if there is a need to change the software, then a new version of the software is created with a new iteration.

The Iterative Model allows the accessing earlier phases, in which the variations made respectively.



WHEN TO USE:

a. When requirements are defined clearly and easy to understand.
b. When the software application is large.
c. When there is a requirement of changes in future.

ADVANTAGES:

a. Testing and debugging during smaller iteration is easy.
b. A Parallel development can plan.

c. It is easily acceptable to ever-changing needs of the project.
d. Risks are identified and resolved during iteration.
e. Limited time spent on documentation and extra time on designing

DISADVANTAGES:

a. It is not suitable for smaller projects.
b. More Resources may be required.
c. Design can be changed again and again because of imperfect requirements.
d. Requirement changes can cause over budgeting.
e. Project completion date not confirmed because of changing requirements.