SOFTWARE PROCESS MODELS

WHAT IS A PROCESS?

- A series of steps taken to produce an intended output
- Steps involves
  - Activities
  - Constraints
  - Resources
- Process involves
  - Tools
  - techniques

Definition of a Process:

Series of Steps: A process consists of a sequence of steps or actions that need to be followed in a particular order to accomplish a defined objective.
Intended Output: The purpose of the process is to produce a desired outcome, which could be a product, service, or result.

Elements of a Process:

Activities: These are the individual tasks or actions that need to be performed within each step of the process. Activities contribute to the completion of the overall objective.
Constraints: Constraints are limitations or conditions that affect how the activities are performed or the resources that can be utilized during the process. They may include time constraints, budget limitations, or technological restrictions.
Resources: Resources refer to the tools, materials, people, and other assets required to execute the activities within the process. Resources are essential for carrying out the necessary work and achieving the desired output.

Components of a Process:

Tools: Tools are the instruments or software applications used to facilitate the execution of activities within the process. They can include software tools, machinery, equipment, or any other aids that assist in performing tasks efficiently.
Techniques: Techniques refer to the methods, procedures, or approaches employed to carry out the activities within the process effectively. These techniques may involve specific methodologies, best practices, or guidelines tailored to the requirements of the process.

Example:

Series of Steps: The process begins with requirements gathering, followed by design, implementation, testing, deployment, and maintenance.
Intended Output: The goal is to produce a functioning software application that meets the specified requirements and satisfies the needs of the users.
Activities: Activities within the process include conducting user interviews, creating design documents, writing code, performing unit tests, deploying the software, and providing ongoing support.
Constraints: Constraints might include deadlines for project completion, budget limitations, compatibility requirements with existing systems, and adherence to regulatory standards.
Resources: Resources necessary for the software development process may include skilled developers, project management software, programming languages, testing tools, hardware infrastructure, and funding.
Tools: Examples of tools used in the software development process include integrated development environments (IDEs), version control systems, bug tracking software, and automated testing frameworks.
Techniques: Techniques such as Agile development, Scrum methodology, Test-Driven Development (TDD), and Continuous Integration (CI) may be employed to enhance productivity, quality, and collaboration throughout the software development lifecycle.

CHARACTERISTICS OF PROCESS

- Prescribes all major process activities
- Uses resources, subject to set of constraints (such as schedule, no. of people working)
- Produces intermediate and final products
- May be composed of sub-processes with hierarchy or links
- Each process activity has entry and exit criteria
- Activities are organized in sequence, so timing is clear
- Each process guiding principles, including goals of each activity
- Constraints may apply to an activity, resource or product

WHY ARE PROCESSES SO IMPORTANT

- Impose consistency and structure on a set of activities
- Guide us to understand, control, examine, and improve the activities
- Enable us to capture our experiences and pass them along

WHAT IS MEANT BY MODELING A PROCESS?

- Description of a process at a given level
- A process model is
  - an anticipation of what the process will look like
  - What the process shall be (actually it will be determined during actual system development).

REASONS FOR MODELING A PROCESS

- To form a common understanding
- To find inconsistencies, redundancies, omissions
- To find and evaluate appropriate activities for reaching process goals
- To tailor a general process for a particular situation in which it will be used

SOFTWARE PROCESS MODELS (SPMs): SDLC REVISITED

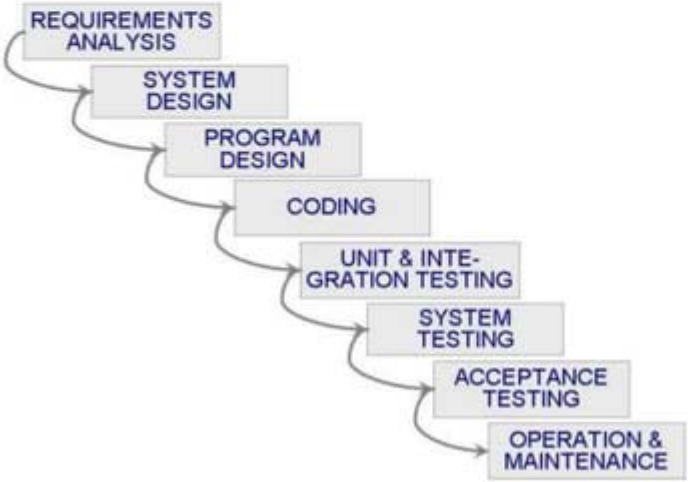
- When a process involves building a software, the process may be referred to as software life cycle
  - Requirements analysis and definition
  - System (architecture) design
  - Program (detailed/procedural) design
  - Writing programs (coding/implementation)
  - Testing: unit, integration, system (many more..)
  - System delivery (deployment)
  - Maintenance

SOME COMMON SPMs

- Waterfall
- Prototyping
- V-Model
- Incremental
- Iterative
- Spiral
- RUP
- Agile Development

WATERFALL MODEL

- one of the first process models proposed. The SDLC is represented as follows by this model



WHEN SHOULD I CHOOSE WATERFALL MODEL AS MY SPM

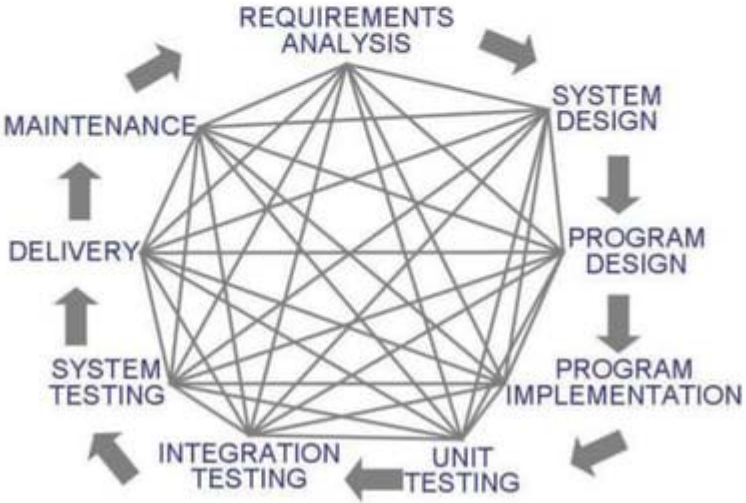
- Works for well understood problems with minimal or no changes in the requirements

ADVANTAGES OF WATERFALL MODEL

- Simple and easy to explain to customers
- It presents
  - a very high-level view of the development process
  - sequence of process activities
- Each major phase is marked by milestones and deliverables (artefacts)

DRAWBACKS OF WATERFALL MODEL

- There is no iteration in waterfall model
- Most software developments apply a great many iterations



DRAWBACKS OF WATERFALL MODEL

- There is no iteration in waterfall model
- Most software developments apply a great many iterations
- Provides no guidance how to handle changes to products and activities during development (assumes requirements can be frozen)
- Views software development as manufacturing process rather than as creative process
- There is no iterative activities that lead to creating a final product
- Long wait before a final product

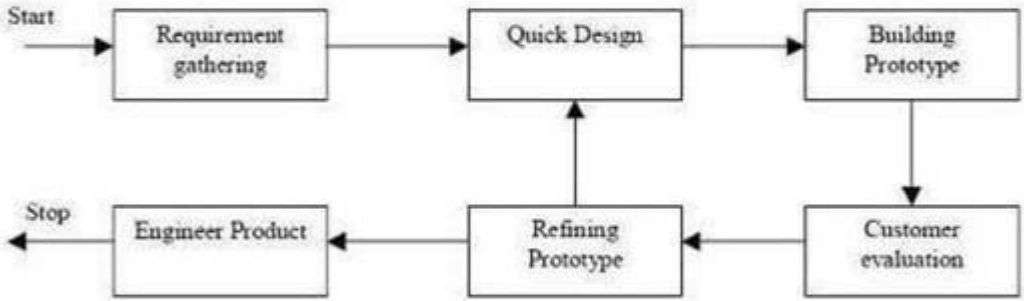
WATERFALL PROCESS MODEL

- Waterfall model is hardly ever used as a solo process model.
- Most of the time it is used along side prototype process model

PROTOTYPE PROCESS MODEL

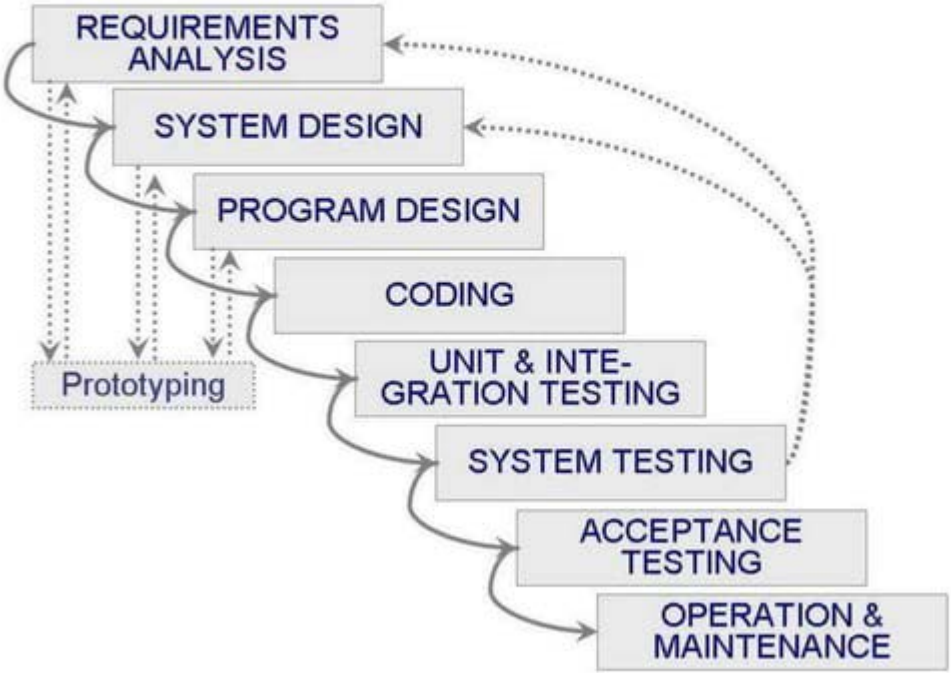
- A prototype is a partially developed product
- Prototyping helps
  - developers assess alternative design strategies (design prototype)
  - users understand what the system will be like (user interface prototype)
- Prototyping is useful for verification and validation

TYPICAL REPRESENTATION OF PROTOTYPE MODEL



Prototyping Model

WATERFALL MODEL WITH PROTOTYPE



ADVANTAGES OF PROTOTYPE PROCESS MODEL

- Interaction of Users and stakeholders at there respective phases
- working model→ better understanding.
- Errors can be detected much earlier.
- Quicker user feedback →better solutions.
- Missing, ambiguous , complex functionality can be identified easily
- Requirements validation, Quick implementation of, incomplete, but functional, application.

DISADVANTAGES OF PROTOTYPE PROCESS MODEL

- Leads to implementing and then repairing way of building systems.
- Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
- Incomplete application may cause application not to be used as the full system was designed
  Incomplete or inadequate problem analysis.

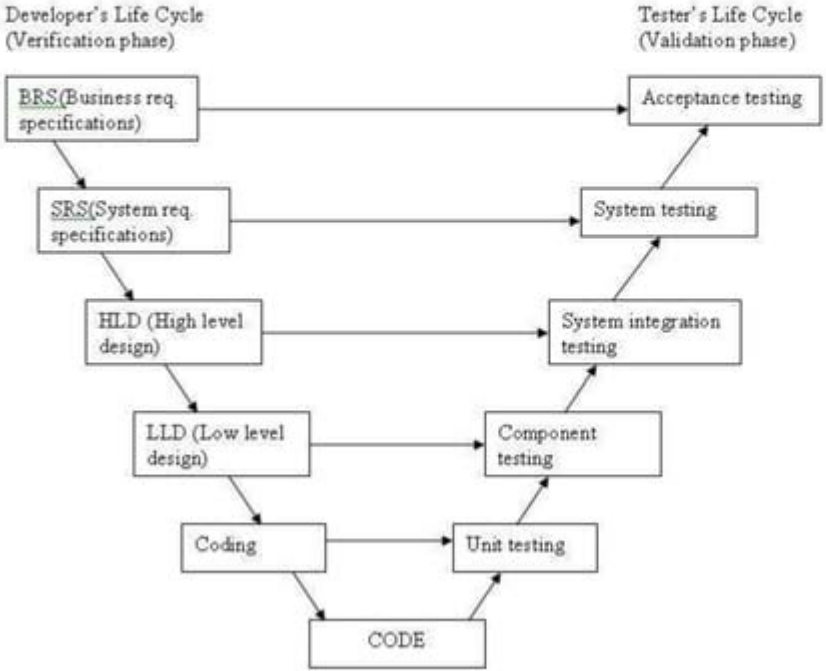
WHEN TO USE PROTOTYPE MODEL

- desired system →lot of user interaction
- E.g online systems, web interfaces
- end users constantly work with the system →feedback →incorporated in the prototype →useable system.
- excellent for designing good human computer interface systems.

V MODEL

- A variation of the waterfall model
- Uses unit testing to verify procedural design
- Uses integration testing to verify architectural (system) design
- Uses acceptance testing to validate the requirements
- If problems are found during verification and validation, the left side of the V can be re-executed before testing on the right side is re-enacted

REPRESENTATION OF V PROCESS MODEL



PHASES OF V MODEL

- **Requirements** like BRS and SRS begin the life cycle model just like the waterfall model. But, in this model before development is started, a system test plan is created. The test plan focuses on meeting the functionality specified in the requirements gathering.
- **The high-level design (HLD)** phase focuses on system architecture and design. It provide overview of solution, platform, system, product and service/process. An integration test plan is created in this phase as well in order to test the pieces of the software systems ability to work together.
- **The low-level design (LLD)** phase is where the actual software components are designed. It defines the actual logic for each and every component of the system. Class diagram with all the methods and relation between classes comes under LLD. Component tests are created in this phase as well.
- **The implementation** phase is, again, where all coding takes place. Once coding is complete, the path of execution continues up the right side of the V where the test plans developed earlier are now put to use.
- **Coding:** This is at the bottom of the V-Shape model. Module design is converted into code by developers.

ADVANTAGES OF V MODEL

- Simple and easy to use.
- Testing activities like planning, test designing happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.
- Proactive defect tracking – that is defects are found at early stage.
- Avoids the downward flow of the defects.
- Works well for small projects where requirements are easily understood.

DISADVANTAGE OF V MODEL

- Very rigid and least flexible.
- Software is developed during the implementation phase, so no early prototypes of the software are produced.
- If any changes happen in midway, then the test documents along with requirement documents has to be updated.
- High confidence of customer is required for choosing the V-Shaped model approach.
- Since, no prototypes are produced, there is a very high risk involved in meeting customer expectations.

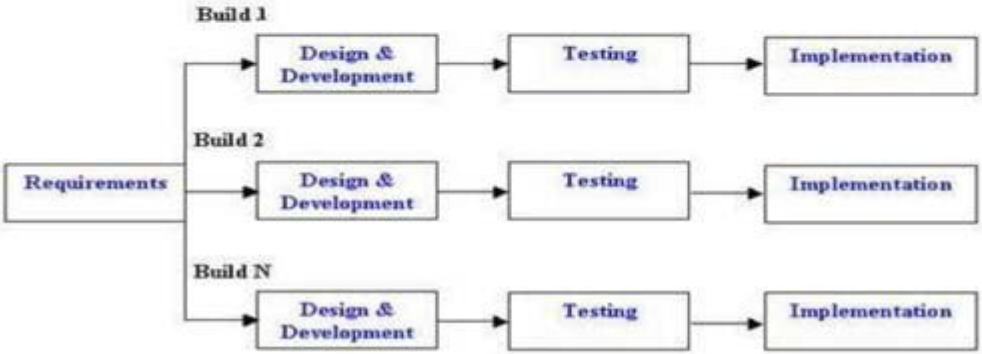
WHEN SHOULD I USE V MODEL

- The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.
- The V-Shaped model should be chosen when ample technical resources are available with needed technical expertise.

INCREMENTAL MODEL

- whole requirement → various builds.
- Multiple development cycles→"multi-waterfall" cycle.
- Cycles → smaller, more easily managed modules.
- Each module → phases (req, design, imp and testing)
- First module → working version of software→ working software early on during the SDLC.
- subsequent release (module) →adds function to the previous release.
- process continues till the complete system is achieved.

REPRESENTATION OF INCREMENTAL PROCESS MODEL



Incremental Life Cycle Model

ADVANTAGES OF INCREMENTAL MODEL

- Generates working software quickly and early during the software life cycle.
- More flexible – less costly to change scope and requirements.
- Easier to test and debug during a smaller iteration.
- Customer can respond to each built.
- Lowers initial delivery cost.
- Easier to manage risk because risky pieces are identified and handled during its iteration.

DISADVANTAGE OF INCREMENTAL MODEL

- Needs good planning and design.
- Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
- Total cost is higher than waterfall.

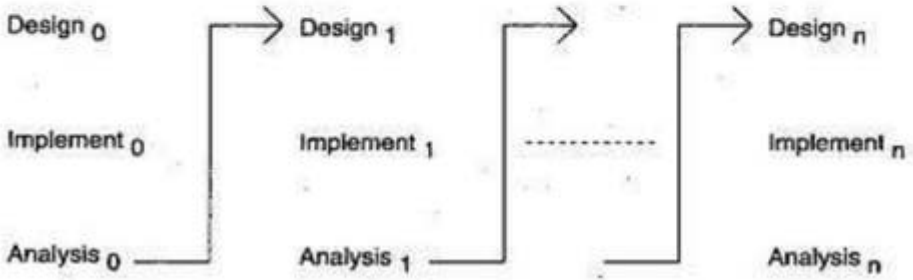
WHEN SHOULD I USE INCREMENTAL MODEL

- Requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some details can evolve with time.
- There is a need to get a product to the market early.
- A new technology is being used
- Resources with needed skill set are not available
- There are some high risk features and goals.

ITERATIVE MODEL

- no start with a full specification of requirements.
- development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements.
- process is then repeated, producing a new version of the software for each cycle of the model.

REPRESENTATION OF ITERATIVE PROCESS MODEL



ADVANTAGES OF ITERATIVE MODEL

- Can only create a high-level design of the application before beginning the actual product and define the design solution for the entire product.
- design and built a skeleton version of that, and then evolved the design based on what had been built.
- building and improving the product step by step.
- track the defects at early stages → avoids the downward flow of the defects.
- reliable user feedback. When presenting sketches and blueprints of the product to users for their feedback, we are effectively asking them to imagine how the product will work.
- In iterative model less time is spent on documenting and more time is given for designing.

DISADVANTAGE OF ITERATIVE MODEL

- Each phase of an iteration is rigid with no overlaps
- Costly system architecture or design issues may arise because not all requirements are gathered up front for the entire lifecycle

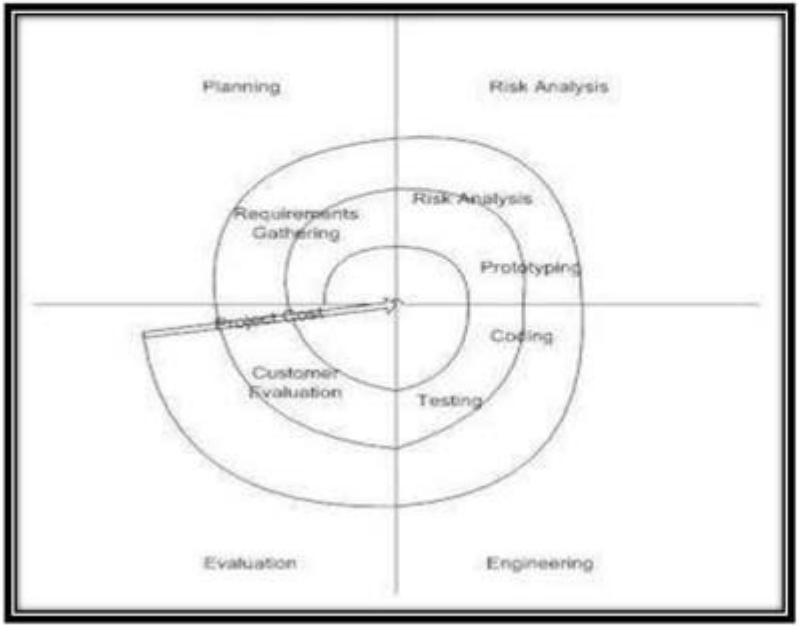
WHEN SHOULD I USE ITERATIVE MODEL

- Requirements of the complete system are clearly defined and understood.
- When the project is big.
- Major requirements must be defined; however, some details can evolve with time.

SPIRAL MODEL

- Similar to incremental model
- More focus on risk analysis
- The spiral model has four phases:
  - Planning
  - Engineering (Determine goals, alternatives and constraints)
  - Risk Analysis (Evaluate alternatives and risks)
  - Evaluation (Develop and test)
- A prototype is produced at the end of the risk analysis phase.

REPRESENTATION OF SPIRAL PROCESS MODEL



ADVANTAGES OF SPIRAL MODEL

- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Additional Functionality can be added at a later date.
- Software is produced early in the software life cycle.

DISADVANTAGE OF SPIRAL MODEL

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

WHEN SHOULD I USE SPIRAL MODEL

- When costs and risk evaluation is important
- For medium to high-risk projects
- Long-term project commitment unwise because of potential changes to economic priorities
- Users are unsure of their needs
- Requirements are complex
- New product line
- Significant changes are expected (research and exploration)

RDP MODEL

- RDP model is Rapid Application Development model.
- type of incremental model
- components or functions are developed in parallel as if they were mini projects.
- developments are time boxed, delivered and then assembled into a working prototype
- quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.

PHASES IN RDP MODEL

- **Business modeling:** information flow between various business functions.
- **Data modeling:** Information flow→ data objects
- **Process modeling:** Data objects →achieve some specific business objective + Description → CRUD of data objects.
- **Application generation:** Automated tools used, process models → code + actual system.
- **Testing and turnover:** Test new components and all the interfaces.

REPRESENTATION OF RDP PROCESS MODEL

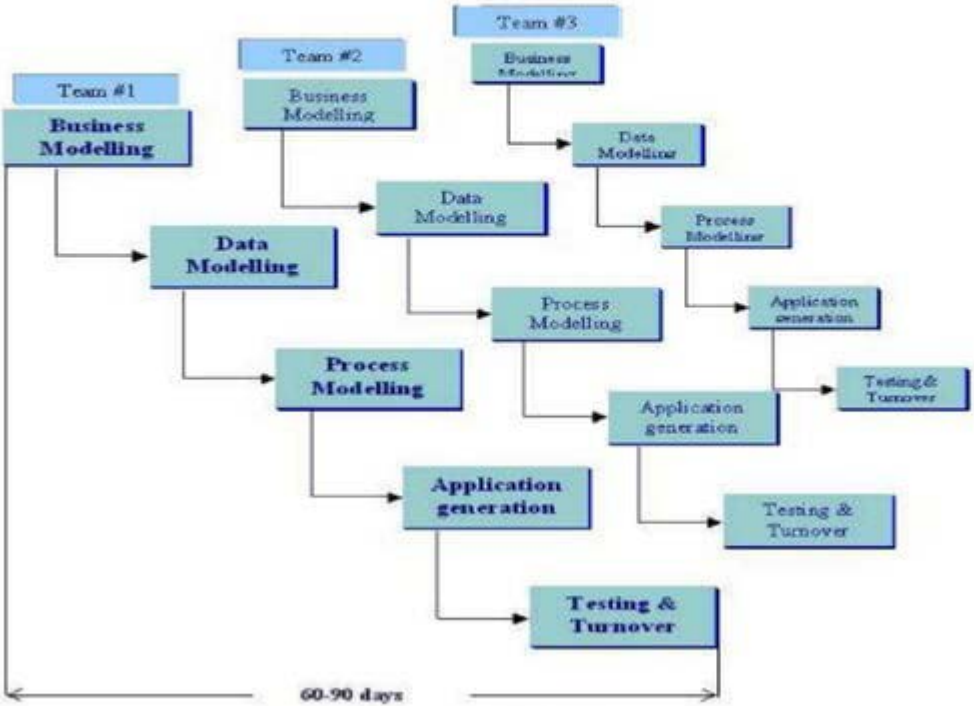


Figure 1.5 – RAD Model

ADVANTAGES OF RDP MODEL

- Reduced development time.
- Increases reusability of components
- Quick initial reviews occur
- Encourages customer feedback
- Integration from very beginning solves a lot of integration issues.

DISADVANTAGE OF RDP MODEL

- Depends on strong team and individual performances for identifying business requirements.
- Only system that can be modularized can be built using RAD
- Requires highly skilled developers/designers.
- High dependency on modeling skills
- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

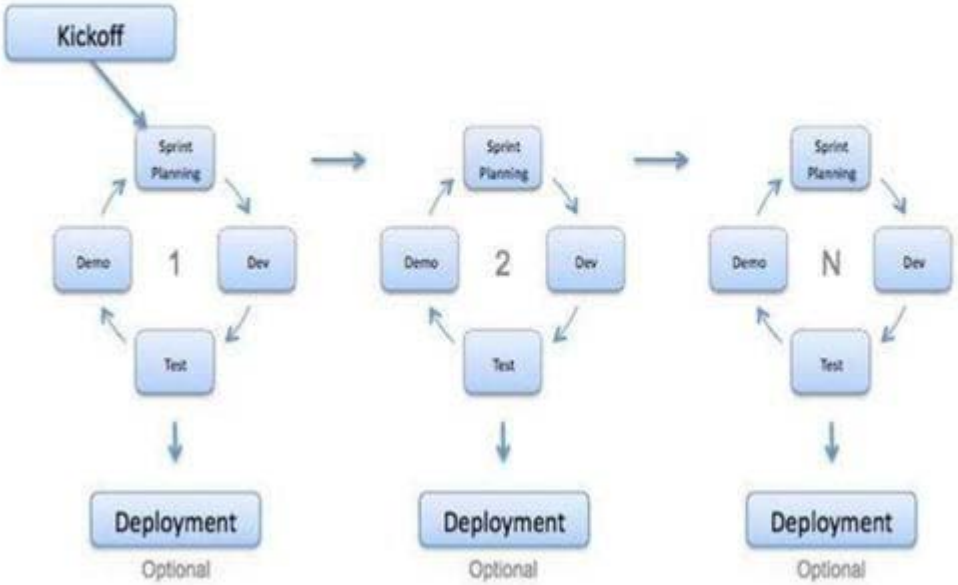
WHEN SHOULD I USE RDP MODEL

- a need to create a system that can be modularized in 2-3 months of time.
- there's high availability of designers for modeling
- budget is high enough
  - cost of designers
  - cost of automated code generating tools.
- resources with high business knowledge are available
- short span of time (2-3 months).

AGILE MODEL

- a type of Incremental model.
- Software is developed in incremental, rapid cycles →small incremental releases with each release building on previous functionality.
- Each release is thoroughly tested to ensure software quality is maintained. It is used for time critical applications.
- Extreme Programming (XP) is currently one of the most well known agile development life cycle model.

REPRESENTATION OF AGILE PROCESS MODEL



ADVANTAGES OF AGILE MODEL

- Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication.
- Close, daily cooperation between business people and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed

DISADVANTAGE OF AGILE MODEL

- In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
- Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.

WHEN SHOULD I USE AGILE MODEL

- When new changes are needed to be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.
- To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.
- Unlike the waterfall model in agile model very limited planning is required to get started with the project. Agile assumes that the end users' needs are ever changing in a dynamic business and IT world. Changes can be discussed and features can be newly effected or removed based on feedback. This effectively gives the customer the finished system they want or need.
- Both system developers and stakeholders alike, find they also get more freedom of time and options than if the software was developed in a more rigid sequential way. Having options gives them the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project can continue to move forward without fear of reaching a sudden standstill.