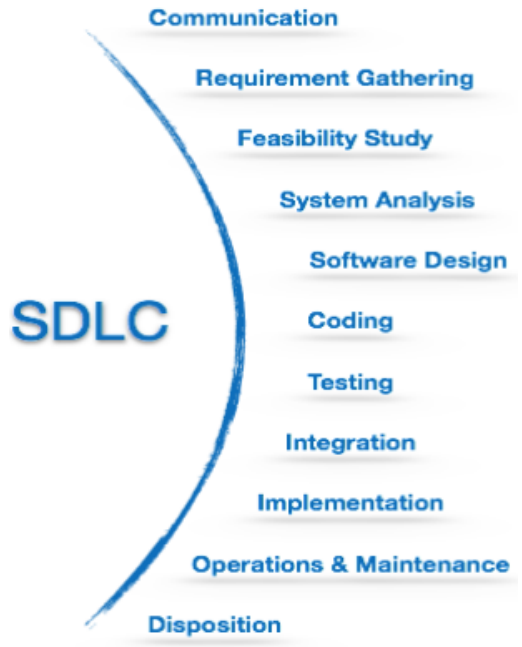
Software Development Life Cycle

Software Development Life Cycle

Software Development Life Cycle, SDLC for short, is a well-defined, structured sequence of stages in software engineering to develop the intended software product.

SDLC provides a series of steps to be followed to design and develop a software product efficiently. SDLC framework includes the following steps:

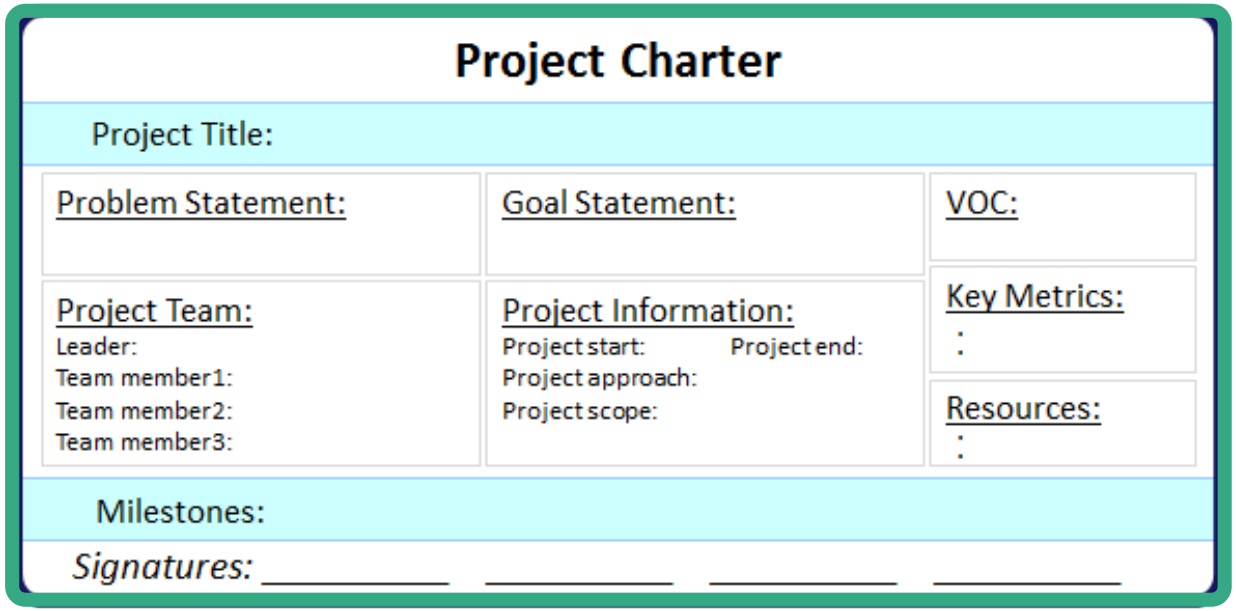


- **Communication**
  - This is the first step where the user initiates the request for a desired software product. The user contacts the service provider and tries to negotiate the terms, submits the request to the service providing organization in writing.

Software Development Life Cycle

- **Requirement Gathering**
  - This step onwards the software development team works to carry on the project. The team holds discussions with various stakeholders from problem domain and tries to bring out as much information as possible on their requirements. The requirements are contemplated and segregated into user requirements, system requirements and functional requirements. The requirements are collected using a number of practices as given –
    - studying the existing or obsolete system and software,
    - conducting interviews of users and developers,
    - referring to the database or
    - collecting answers from the questionnaires.

Software Development Life Cycle



Software Development Life Cycle

- **Feasibility Study**
  - After requirement gathering, the team comes up with a rough plan of software process. At this step the team analyses if a software can be designed to fulfil all requirements of the user, and if there is any possibility of software being no more useful. It is also analysed if the project is financially, practically, and technologically feasible for the organization to take up. There are many algorithms available, which help the developers to conclude the feasibility of a software project.

- **System Analysis**
  - At this step the developers decide a roadmap of their plan and try to bring up the best software model suitable for the project. System analysis includes understanding of software product limitations, learning system related problems or changes to be done in existing systems beforehand, identifying and addressing the impact of project on organization and personnel etc. The project team analyses the scope of the project and plans the schedule and resources accordingly.

- **Software Design**
  - Next step is to bring down whole knowledge of requirements and analysis on the desk and design the software product. The inputs from users and information gathered in requirement gathering phase are the inputs of this step. The output of this step comes in the form of two designs; logical design, and physical design. Engineers produce meta-data and data dictionaries, logical diagrams, data-flow diagrams, and in some cases pseudo codes.

- **Coding**
  - This step is also known as programming phase. The implementation of software design starts in terms of writing program code in the suitable programming language and developing error-free executable programs efficiently.

- **Testing**
  - An estimate says that 50% of whole software development process should be tested. Errors may ruin the software from critical level to its own removal. Software testing is done while coding by the developers and thorough testing is conducted by testing experts at various levels of code such as module testing, program testing, product testing, in-house testing, and testing the product at user's end. Early discovery of errors and their remedy is the key to reliable software.

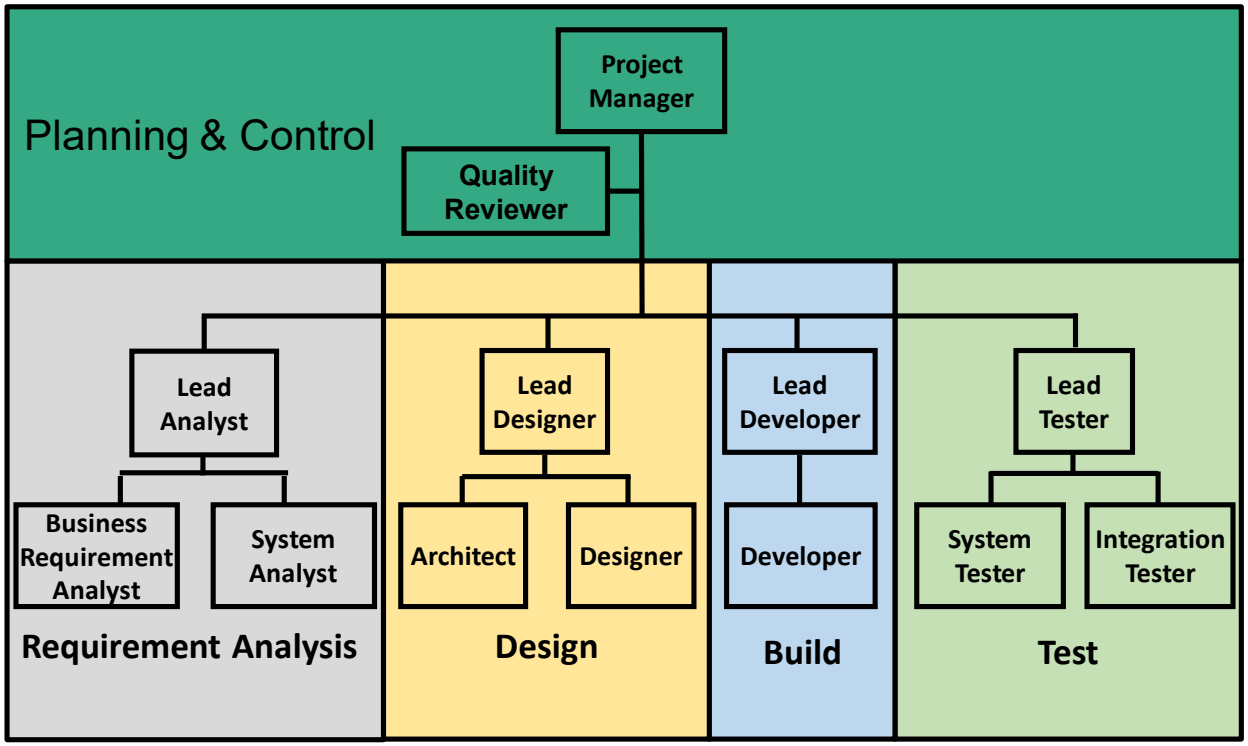
- **Integration**
  - Software may need to be integrated with the libraries, databases, and other program(s). This stage of SDLC is involved in the integration of software with outer world entities.

- **Implementation**
  - This means installing the software on user machines. At times, software needs post-installation configurations at user end. Software is tested for portability and adaptability and integration related issues are solved during implementation.

Software Development Life Cycle

- **Operation and Maintenance**
  - This phase confirms the software operation in terms of more efficiency and less errors. If required, the users are trained on, or aided with the documentation on how to operate the software and how to keep the software operational. The software is maintained timely by updating the code according to the changes taking place in user end environment or technology. This phase may face challenges from hidden bugs and real-world unidentified problems.

Roles Associated with Various Phases in SDLC



Software Development Paradigm

Software Development Paradigm

- The software development paradigm helps a developer to select a strategy to develop the software. A software development paradigm has its own set of tools, methods, and procedures, which are expressed clearly and defines software development life cycle. A few of software development paradigms or process models are defined as follows:
  - Waterfall Model
  - Structured Evolutionary Prototyping Model
  - Incremental Model
  - Rapid Application Development (RAD) Model
  - Spiral Model
  - V-Model
  - Scrum Development Model
  - Big Bang Model

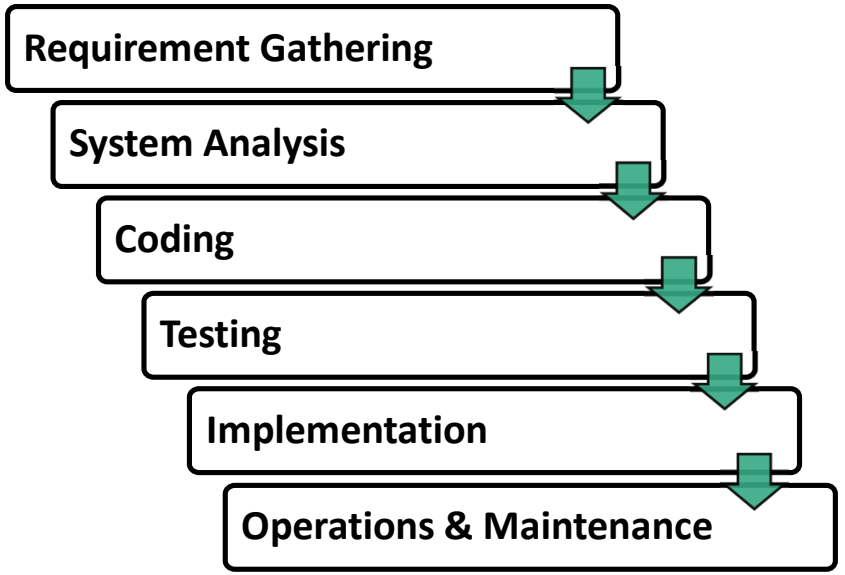
Etc.

Waterfall Model

Waterfall Model

Waterfall model is the simplest model of software development paradigm. All the phases of SDLC will function one after another in linear manner. That is, when the first phase is finished then only the second phase will start and so on.

Waterfall Model

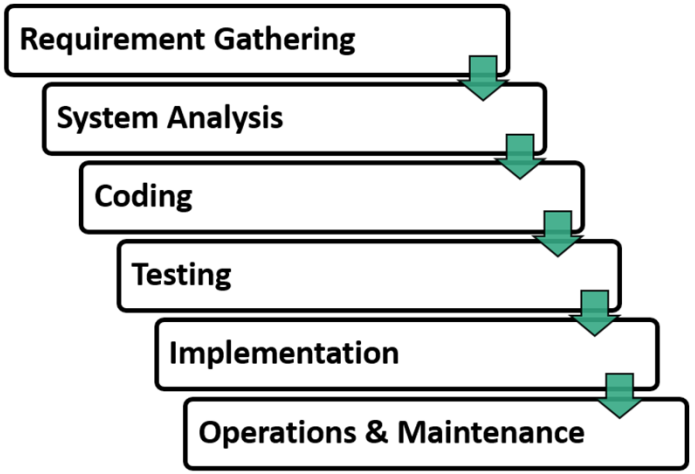


Waterfall Model

This model assumes that everything is carried out and taken place perfectly as planned in the previous stage and there is no need to think about the past issues that may arise in the next phase. This model does not work smoothly if there are some issues left at the previous step. The sequential nature of model does not allow us to go back and undo or redo our actions.

This model is best suited when developers already have designed and developed similar software in the past and are aware of all its domains.

Waterfall Model



Waterfall Strengths

- Ease of Use: Easy or difficult ?
  - Easy to understand, easy to use
  - Provides structure to inexperienced staff
- Clarity on Milestones: Clear or not?
  - Well understood
- Importance is given to quality or schedule?
  - Works well when quality is more important than cost or schedule

Waterfall Deficiencies

- Can new requirements be taken into consideration during the development phase?
  - All requirements must be known upfront
- Can deliverables be subjected to change?
  - Deliverables created for each phase are considered frozen – inhibits flexibility
- Can the customer preview the system during the development phase?
  - Little opportunity for customer to preview the system (until it may be too late)

When to Use the Waterfall Model

- Requirements are very well known
- Product definition is stable
- Technology is understood
- New version of an existing product
- Porting an existing product to a new platform

Structured Evolutionary Prototyping Model

Structured Evolutionary Prototyping Model

- Developers build a prototype during the requirements phase
- Prototype is evaluated by end users
- Users give corrective feedback
- Developers further refine the prototype
- When the user is satisfied, the prototype code is brought up to the standards needed for a final product

Structured Evolutionary Prototyping Steps

- A preliminary project plan is developed
- A partial high-level paper model is created
- The model is source for a partial requirements specification
- A prototype is built with basic and critical attributes

➢ The designer builds
  - ✓ The Database
  - ✓ User Interface
  - ✓ Algorithmic Functions

- The designer demonstrates the prototype, the user evaluates for problems and suggests improvements
- This loop continues until the user is satisfied

Structured Evolutionary Prototyping Strengths

- How does Evolutionary Prototyping Help Customers and Developers?
  - Customers can “see” the system requirements as they are being gathered
  - Developers learn from customers
- How are new requirements dealt with?
  - Unexpected requirements accommodated
  - Allows for flexible design and development
- How can additional needed functionality be identified?
  - Interaction with the prototype stimulates awareness of additional needed functionality

Structured Evolutionary Prototyping Weaknesses

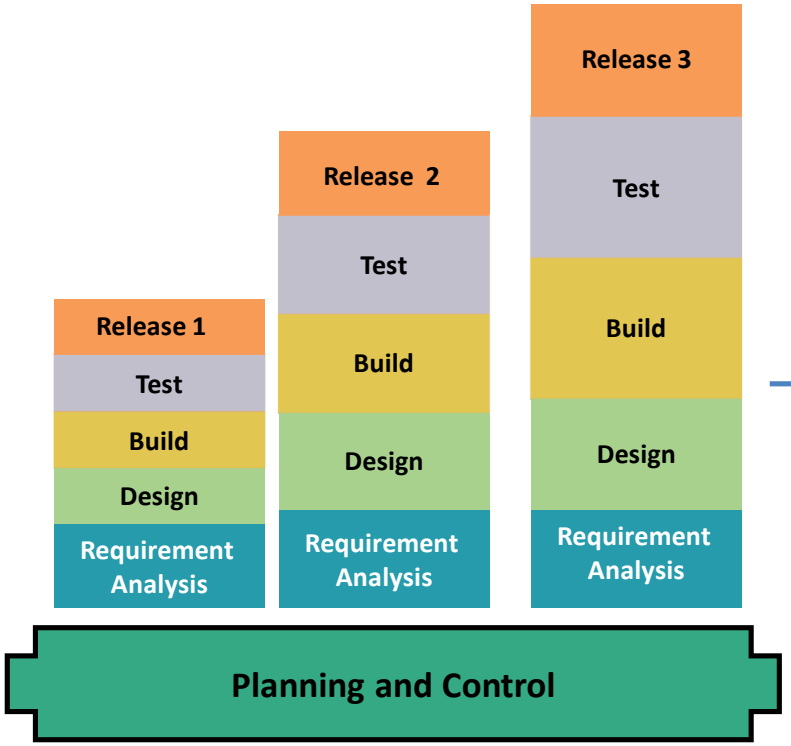
- What is risk involved with this model?
  - Overall maintainability may be overlooked
  - Process may continue forever

When to use Structured Evolutionary Prototyping?

- Requirements are unstable or have to be clarified
- As the requirements clarification stage of a waterfall model
- Develop user interfaces
- Short-lived demonstrations
- New, original development
- With the analysis and design portions of object-oriented development

Incremental Model

Incremental Model



Each release delivers an operational product

Incremental Model Strengths

- What are developed first?
  - Develop high-risk or major functions first
- What does each release deliver?
  - Each release delivers an operational product
- Can Customers use each build?
  - Customer can respond to each build
- Is there any time constraint in product delivery?
  - Initial product delivery is faster

Incremental Model Weaknesses

- What is required early in this model?
  - Requires early definition of a complete and fully functional system to allow for the definition of increments
- How does this model deal with the cost of the complete system?
  - Total cost of the complete system is high

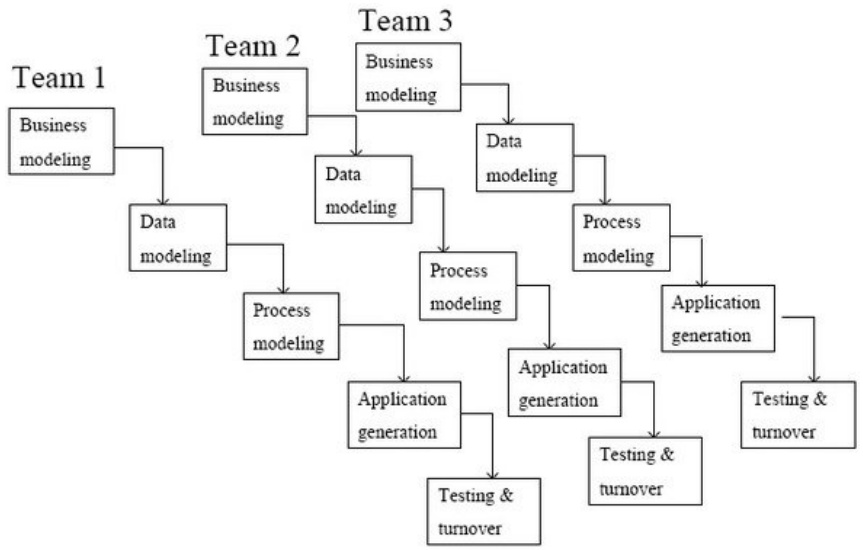
When to Use the Incremental Model?

- When staffing is unavailable for a complete implementation by the business deadline
- Most of the requirements are known up-front but are expected to evolve over time
- A need to get basic functionality to the market early
- On projects which have lengthy development schedules
- On a project with new technology

Rapid Application Development (RAD)

Rapid Application Development (RAD)

- Rapid Application Development (RAD) is a software development methodology that focuses on building applications in a very short amount of time
- It is a high speed adaptation of the linear sequential model in which rapid development is achieved by using component based construction



Core Elements of RAD

- Business Modeling
- Data Modeling
- Process Modeling
- Application Generation
- Testing and Turnover
- RAD tools (Erwin, CASE tools, Rational Rose, Visio)

RAD Strengths

- What mainly gets reduced by using RAD model?
  - Reduced cycle time and improved productivity
- How does customer involvement in the SDLC cycle help?
  - Customer is involved throughout the complete cycle minimizes risk of not achieving customer satisfaction and business needs

RAD Weaknesses

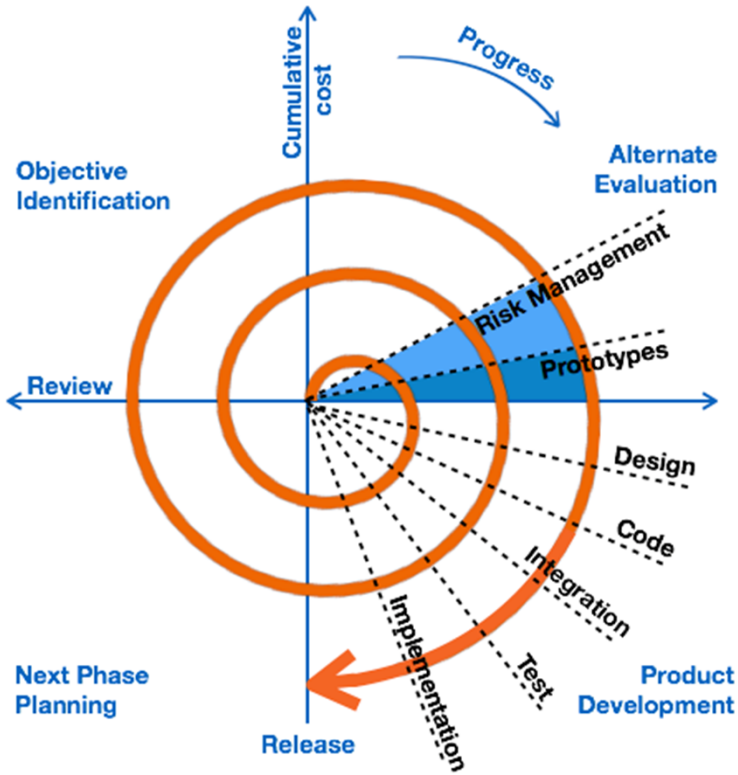
- What is the risk associated with this model?
  - RAD requires sufficient human resources to create the right number of RAD team
  - RAD is not appropriate where technical risk is high
- What is the drawback of customer being involved in the development cycle?
  - Developers and customers must be committed to rapid-fire activities in an abbreviated time frame

When to Use RAD?

- Reasonably well-known requirements
- User involved throughout the life cycle
- Project can be time-boxed
- High performance not required
- Low technical risks
- System can be modularized

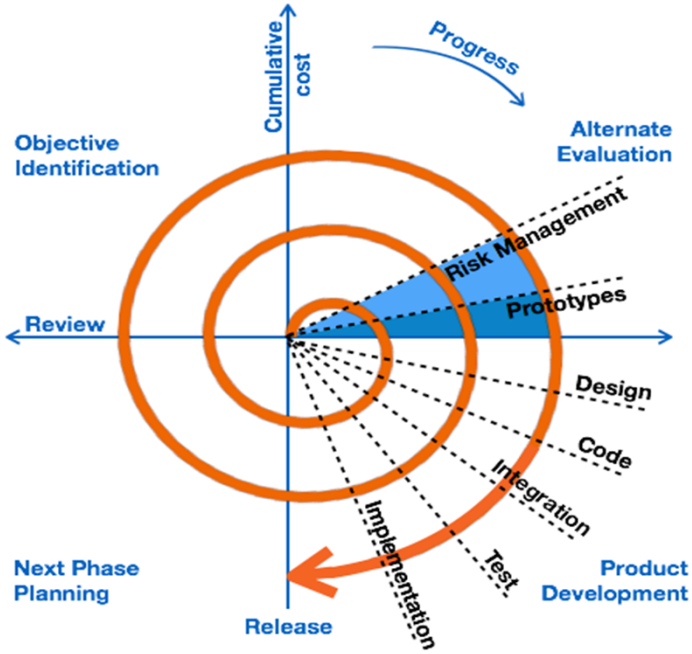
Spiral Model

Spiral Model



Spiral model is a combination of both, iterative model and one of the SDLC models. It can be seen as if you choose one SDLC model and combined it with cyclic process (iterative model).

Spiral Model



This model considers risk, which often goes un-noticed by most other models. The model starts with determining objectives and constraints of the software at the start of one iteration. Next phase is of prototyping the software. This includes risk analysis. Then one standard SDLC model is used to build the software. In the fourth phase of the plan of next iteration is prepared.

Spiral Model Strengths

- What is developed first?
  - Critical high-risk functions are developed first
  - This model provides early indication of existence of risks, without much cost
- How does the user benefit?
  - Users can be closely tied to all lifecycle steps
  - Users see the system early because of rapid prototyping tools
  - Early and frequent feedback from users

Spiral Model Weaknesses

- What is the impact of using Spiral Model for small or low risk Projects?
  - Time spent for evaluating risks too large for small or low-risk projects
  - Time spent planning, resetting objectives, doing risk analysis and prototyping may be excessive
- Can Spiral Model be considered Simple?
  - The model is complex
- Expertise is required in which area?
  - Risk assessment expertise is required

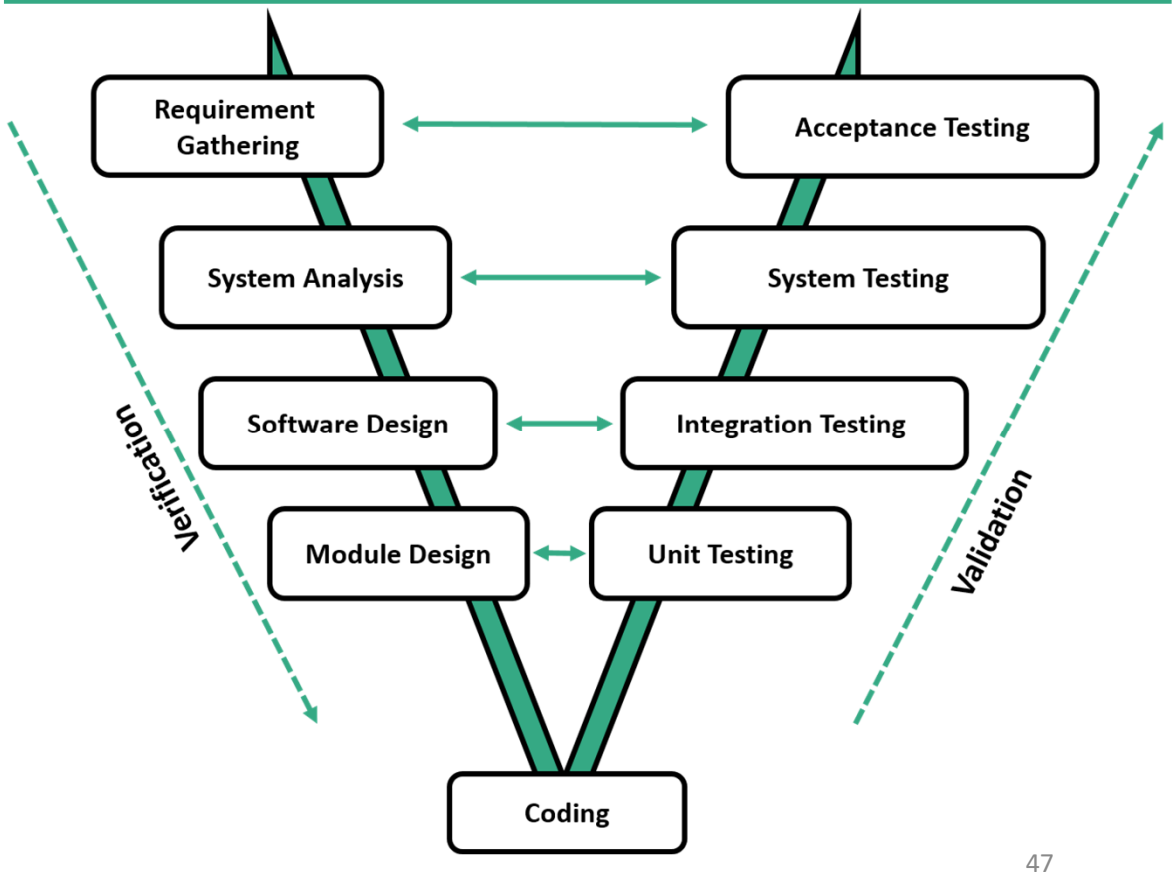
When to Use Spiral Model?

- When creation of a prototype is appropriate
- 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)

V-Model

V-Model

The major drawback of waterfall model is we move to the next stage only when the previous one is finished and there was no chance to go back if something is found wrong in later stages. V-Model provides means of testing of software at each stage in reverse manner.



V-Model

- At every stage, test plans and test cases are created to verify and validate the product according to the requirement of that stage. For example, in requirement gathering stage the test team prepares all the test cases in correspondence to the requirements. Later, when the product is developed and is ready for testing, test cases of this stage verify the software against its validity towards requirements at this stage.
- This makes both verification and validation go in parallel. This model is also known as verification and validation model.

V-Shaped Strengths

- Which phase is more emphasized when using V Shaped Model?
  - Emphasize planning for verification and validation of the product in early stages of product development
  - Each deliverable must be testable – More Emphasis laid on the Testing phase
- Ease of use?
  - Easy to use

V-Shaped Weaknesses

- Can V model handle concurrent events?
  - Does not easily handle concurrent events
- Can V model handle dynamic changes in requirements?
  - Does not easily handle dynamic changes in requirements
- Does V model contain risk analysis activities?
  - Does not contain risk analysis activities

When to use the V-Shaped Model?

- Excellent choice for systems requiring high reliability – hospital patient control applications
- All requirements are known up-front
- When it can be modified to handle changing requirements beyond analysis phase
- Solution and technology are known

KEY DIFFERENCE BETWEEN SCRUM AND AGILE

- Agile is a continuous iteration of development and testing in the software development process whereas Scrum is an Agile process to focus on delivering the business value in the shortest time.
- Agile methodology delivers the software on a regular basis for feedback while Scrum delivers the software after each sprint.
- In the Agile process, leadership plays a vital role; on the other hand, Scrum fosters a self-organizing, cross-functional team.
- Agile involves collaborations and face-to-face interactions between the members of various cross-functional teams whereas Scrum collaboration is achieved in daily stand-up meetings.
- In Agile process design and execution should be kept simple whereas in Scrum process design and execution can be innovative and experimental.

Scrum
Development Model

Scrum Development Model

scrum(rugby)

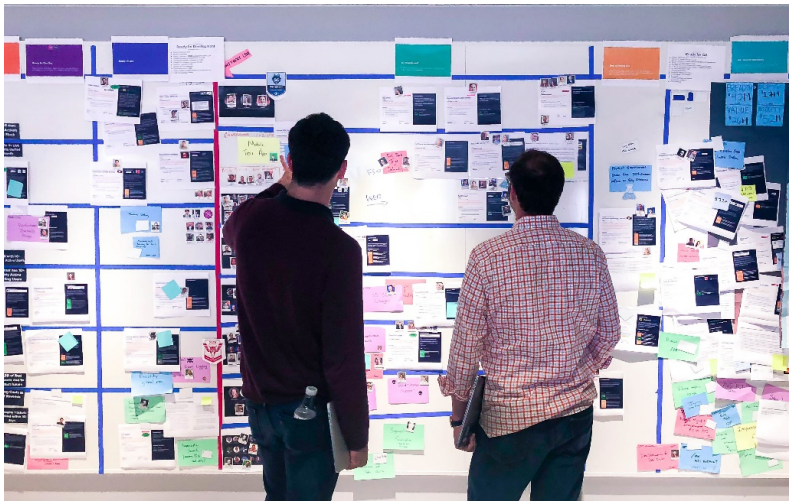
- Originally proposed by Schwaber and Beedle

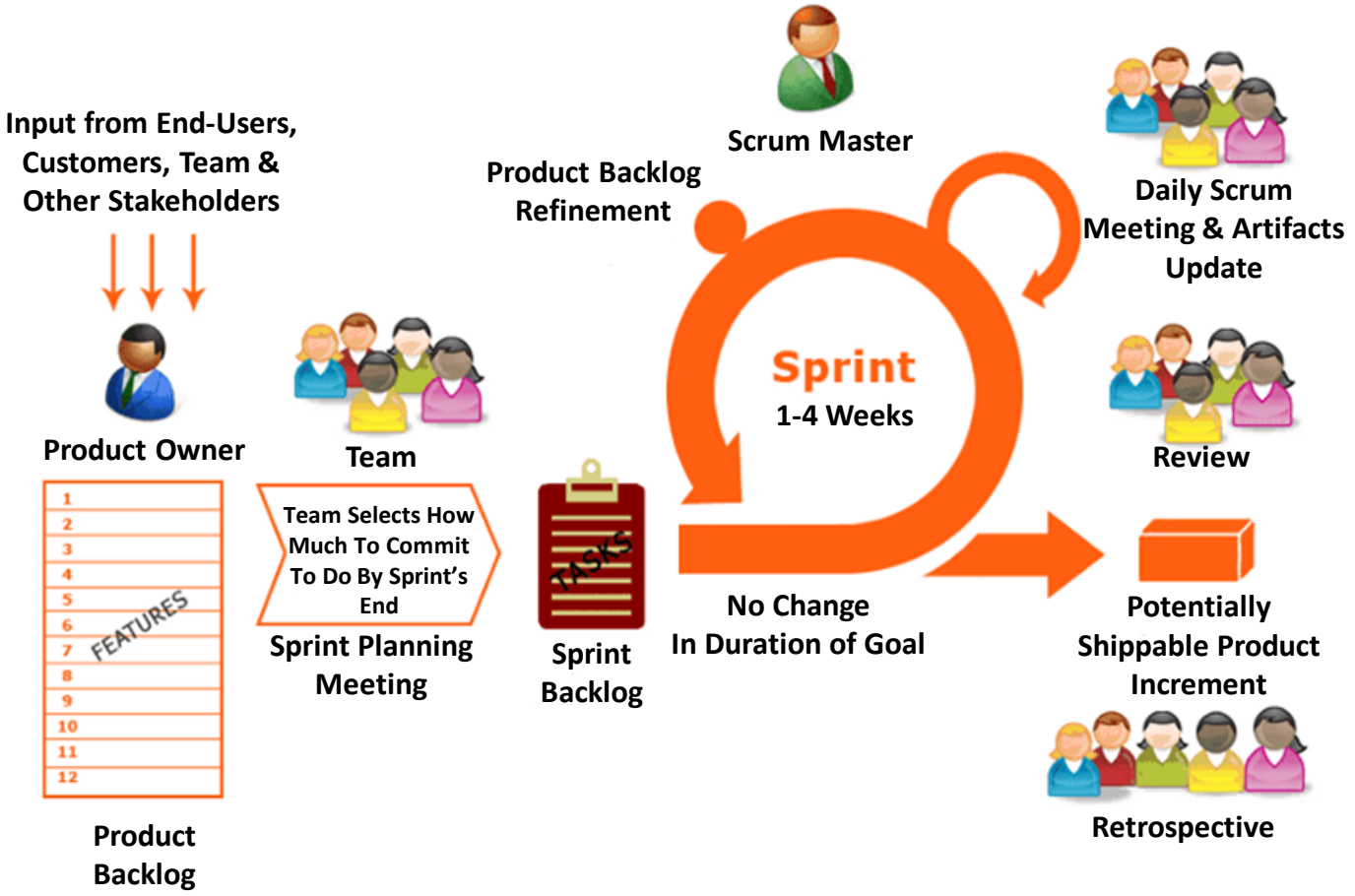
➢ **Scrum—distinguishing features**

- Development work is partitioned into “packets”
- Testing and documentation are on-going as the product is constructed
- Work occurs in “sprints” and is derived from a “backlog” of existing requirements
- Meetings are very short and sometimes conducted without chairs
- “demos” are delivered to the customer with the time-box allocated

Scrum Development Model

Scrum Board





Big Bang Model

Big Bang Model

This model is the simplest model in its form. It requires little planning, lots of programming and lots of funds. This model is conceptualized around the big bang of universe. As scientists say that after big bang lots of galaxies, planets, and stars evolved just as an event. Likewise, if we put together lots of programming and funds, you may achieve the best software product.

Big Bang Model

For this model, very small amount of planning is required. It does not follow any process, or at times the customer is not sure about the requirements and future needs. So the input requirements are arbitrary.

This model is not suitable for large software projects but good one for learning and experimenting.