

Software Development Life Cycle



Compiled by Elizabeth George



SYSTEMS DEVELOPMENT LIFE CYCLE (SDLC)

- ***Systems development life cycle (SDLC)*** - a structured step-by-step approach for developing information systems
- Typical activities include:
 - Determining budgets
 - Gathering business requirements
 - Designing models
 - Writing user documentation

SYSTEMS DEVELOPMENT LIFE CYCLE (SDLC)

SDLC PHASE	ACTIVITIES
1. Planning	<ul style="list-style-type: none">•Define the system to be developed•Set the project scope•Develop the project plan
2. Analysis	<ul style="list-style-type: none">•Gather business requirements
3. Design	<ul style="list-style-type: none">•Design the technical architecture•Design system models
4. Development	<ul style="list-style-type: none">•Build technical architecture•Build databases and programs
5. Testing	<ul style="list-style-type: none">•Write test conditions•Perform testing
6. Implementation	<ul style="list-style-type: none">•Write user documentation•Provide training
7. Maintenance	<ul style="list-style-type: none">•Build a help desk•Support system changes

SYSTEMS DEVELOPMENT LIFE CYCLE (SDLC)



Phase 1: Planning

- ***Planning phase*** - involves determining a solid plan for developing your information system
- Three primary planning activities:
 1. Define the system to be developed
 - ***Critical success factor (CSF)*** - a factor simply critical to your organization' s success

Phase 1: Planning (Requirement Analysis)

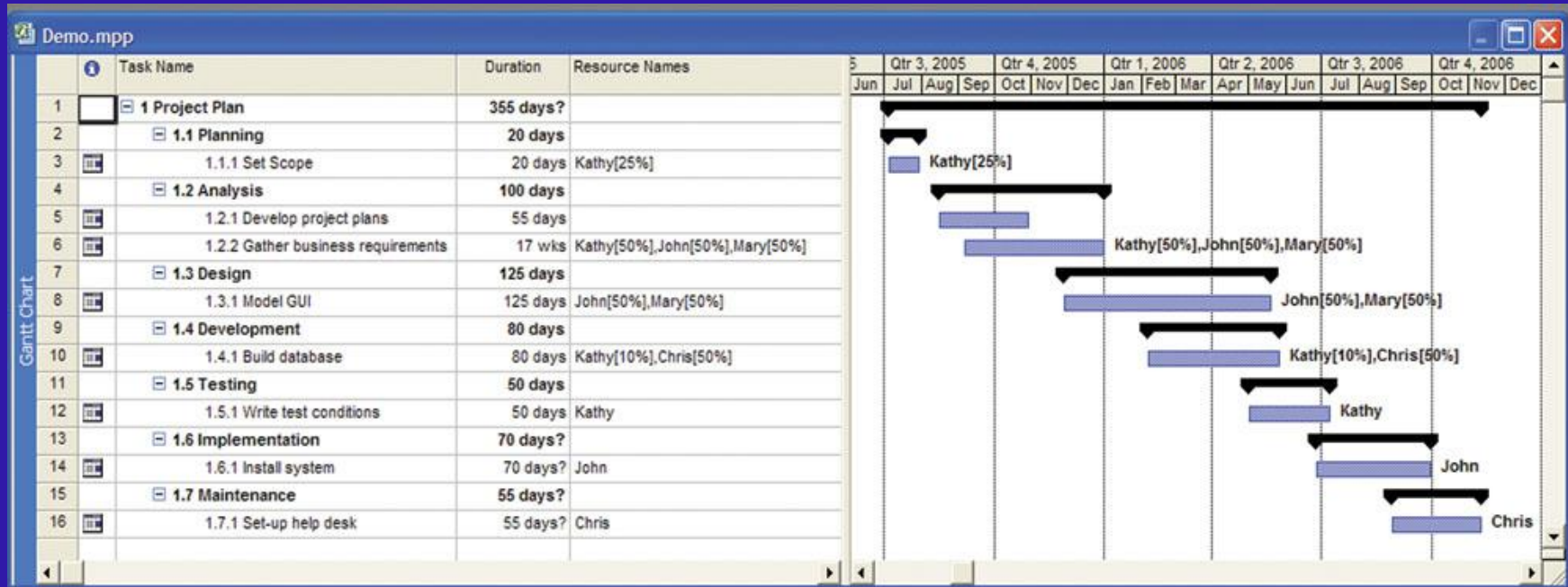
2. Set the project scope

- ***Project scope*** - clearly defines the high-level system requirements
- ***Scope creep*** - occurs when the scope of the project increases
- ***Feature creep*** - occurs when developers add extra features that were not part of the initial requirements
- ***Project scope document*** - a written definition of the project scope and is usually no longer than a paragraph

Phase 1: Planning

3. Develop the project plan including tasks, resources, and timeframes
 - ***Project plan*** - defines the what, when, and who questions of system development
 - ***Project manager*** - an individual who is an expert in project planning and management, defines and develops the project plan and tracks the plan to ensure all key project milestones are completed on time
 - ***Project milestones*** - represent key dates for which you need a certain group of activities performed

Phase 1: Planning



Phase 2: Analysis (Feasibility Study)

- ***Analysis phase*** - involves end users and IT specialists working together to gather, understand, and **document** the business requirements for the proposed system

Phase 2: Analysis

- Two primary analysis activities:
 1. Gather the business requirements
 - ***Business requirements*** - the detailed set of knowledge worker requests that the system must meet in order to be successful
 - ***Joint application development (JAD)*** - knowledge workers and IT specialists meet, sometimes for several days, to define or review the business requirements for the system

Phase 2: Analysis

2. Prioritize the requirements

- ***Requirements definition document*** – prioritizes the business requirements and places them in a formal comprehensive document

Phase 3: Design

- ***Design phase*** - build a technical blueprint of how the proposed system will work
- Two primary design activities:
 1. Design the technical architecture
 - ***Technical architecture*** - defines the hardware, software, and telecommunications equipment required to run the system

Phase 3: Design

2. Design system models

- ***Modeling*** - the activity of drawing a graphical representation of a design
- ***Graphical user interface (GUI)*** - the interface to an information system
- ***GUI screen design*** - the ability to model the information system screens for an entire system

Phase 4: Development

- ***Development phase*** - take all of your detailed design documents from the design phase and transform them into an actual system
- Two primary development activities:
 1. Build the technical architecture
 2. Build the database and programs
 - Both of these activities are mostly performed by IT specialists

Phase 5: Testing

- ***Testing phase*** - verifies that the system works and meets all of the business requirements defined in the analysis phase
- Two primary testing activities:
 1. Write the test conditions
 - ***Test conditions*** - the detailed steps the system must perform along with the expected results of each step

Phase 5: Testing

2. Perform the testing of the system

- ***Unit testing*** – tests individual units of code
- ***System testing*** – verifies that the units of code function correctly when integrated
- ***Integration testing*** – verifies that separate systems work together
- ***User acceptance testing (UAT)*** – determines if the system satisfies the business requirements

Phase 6: Implementation

- ***Implementation phase*** - distribute the system to all of the knowledge workers and they begin using the system to perform their everyday jobs
- Two primary implementation activities
 1. Write detailed user documentation
 - ***User documentation*** - highlights how to use the system

Phase 6: Implementation

2. Provide **training** for the system users
 - ***Online training*** - runs over the Internet or off a CD-ROM
 - ***Workshop training*** - is held in a classroom environment and lead by an instructor

Phase 6: Implementation

- Choose the right implementation method
 - ***Parallel implementation*** – use both the old and new system simultaneously
 - ***Plunge implementation*** – discard the old system completely and use the new
 - ***Pilot implementation*** – start with small groups of people on the new system and gradually add more users
 - ***Phased implementation*** – implement the new system in phases

Phase 7: Maintenance

- ***Maintenance phase*** - monitor and support the new system to ensure it continues to meet the business goals
- Two primary maintenance activities:
 1. Build a help desk to support the system users
 - ***Help desk*** - a group of people who responds to knowledge workers' questions
 2. Provide an environment to support system changes

SDLC Models

- Life cycle models describe the interrelationships between software development phases.
- A simplified representation of software process
- Frameworks that describes the activities performed at each stage of a software development project.

Few of the important SDLC MODELS

1

WATERFALL

2

ITERATIVE

3

V-MODEL

4

SPIRAL

5

LEAN

6

AGILE

7

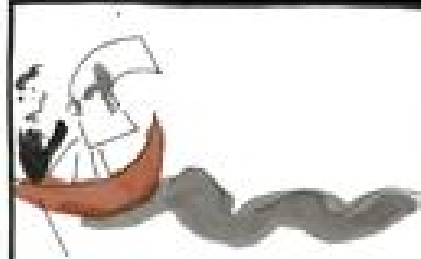
PROTOYPING

8

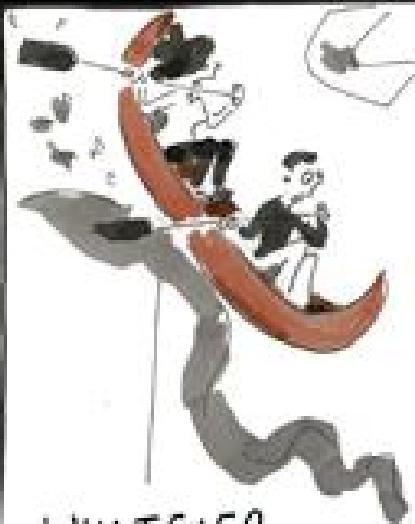
DEVOPS

| 1

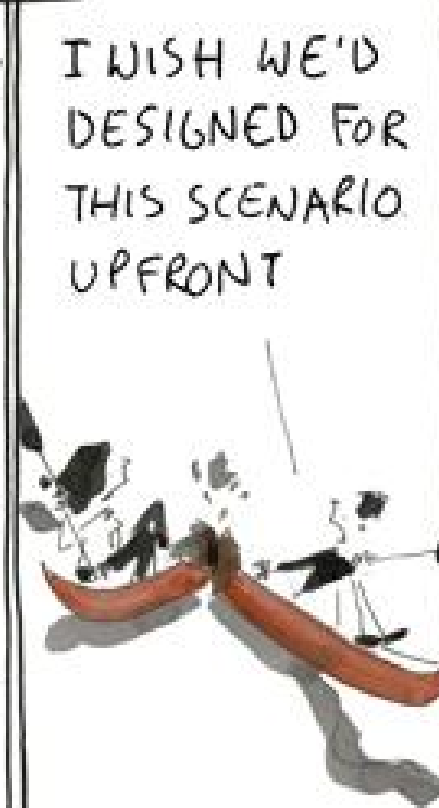
WATERFALL MODEL

THE NEW PRODUCT WATERFALL

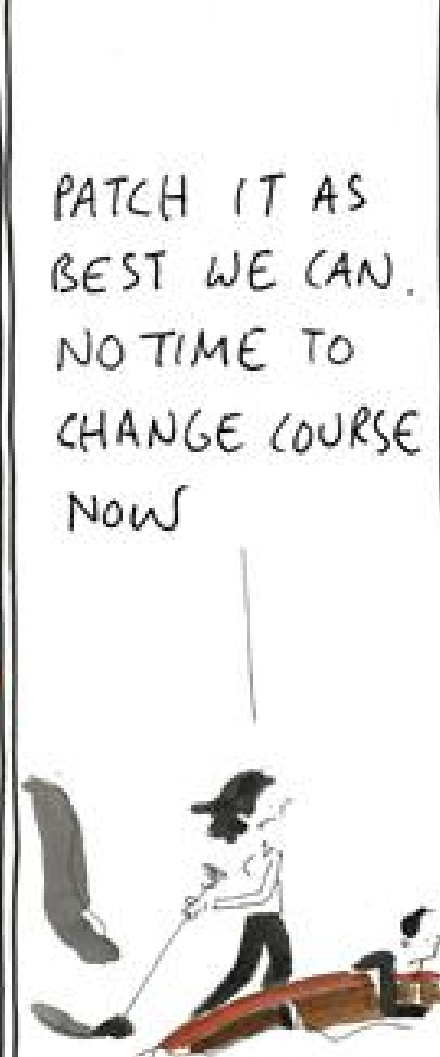
HOW DO WE
CHART OUR
ENTIRE COURSE
IF WE DON'T
KNOW WHAT'S
AHEAD?

PLAN

WHATEVER
HAPPENS, JUST
KEEP PADDLING!

BUILD

I WISH WE'D
DESIGNED FOR
THIS SCENARIO
UPFRONT

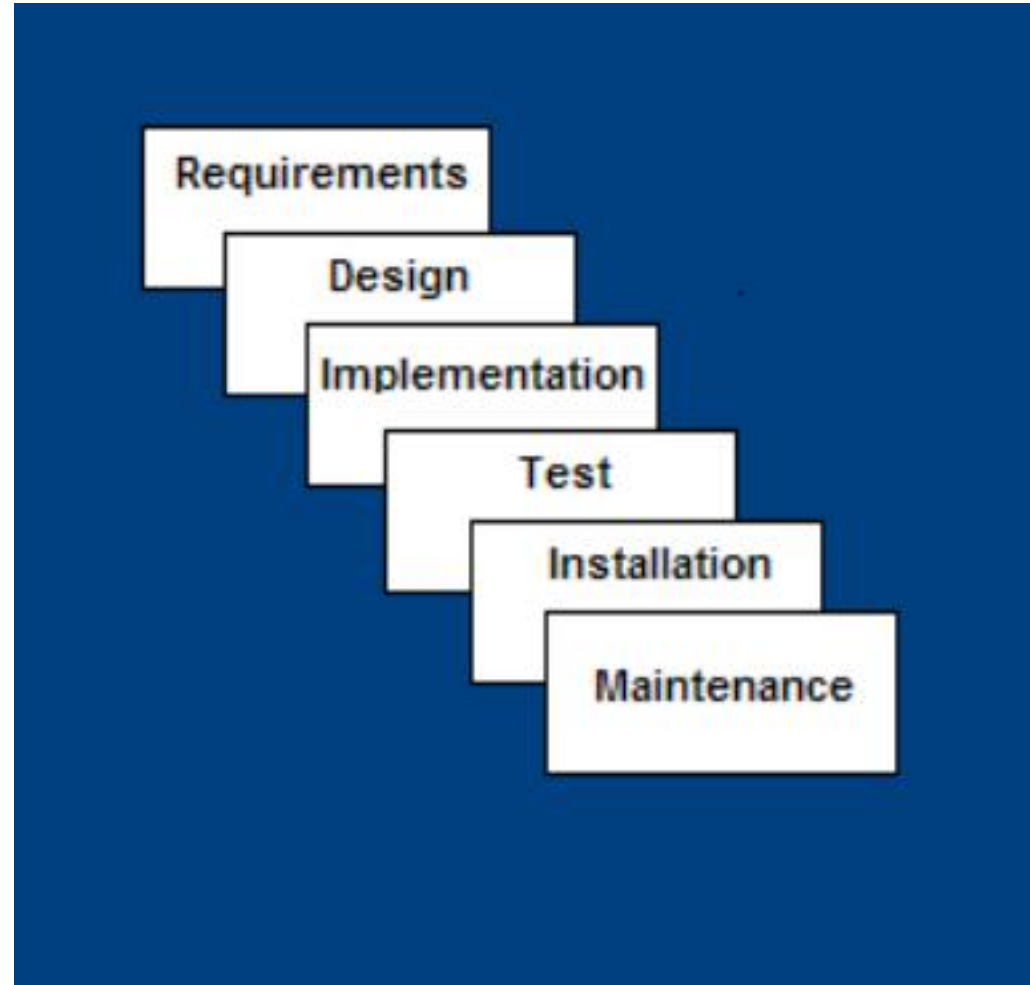
TEST

PATCH IT AS
BEST WE CAN.
NO TIME TO
CHANGE COURSE
NOW

LAUNCH

1. Waterfall Methodology

Waterfall methodology - a sequential, activity-based process

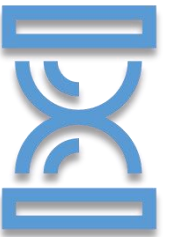
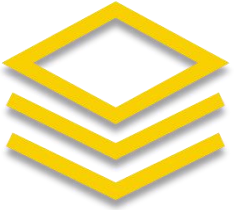


Waterfall Model (contd.)

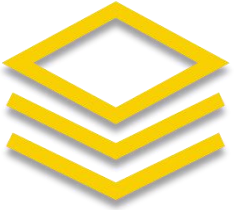
- **The least complicated and most obsolete of the life cycle models. Well suited to projects that has low risk in the areas of user interface and performance requirements, but high risk in budget and schedule predictability and control.**
- Requirements are very well known, clear & fixed.
- Product definition is stable.
- Technology is understood.
- The project is short.

Waterfall Strengths

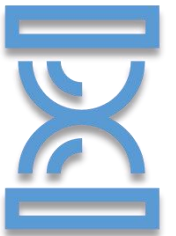
- Easy to understand, easy to use
- Provides structure to inexperienced staff
- Milestones are well understood
- Sets requirements stability
- Good for management control (plan, staff, track)
- Works well when quality is more important than cost or schedule



Waterfall Deficiencies



- All requirements must be known upfront
- Deliverables created for each phase are considered frozen – inhibits flexibility
- Can give a false impression of progress
- Does not reflect problem-solving nature of software development – iterations of phases
- Integration is one big bang at the end
- Little opportunity for customer to preview the system (until it may be too late)



| 2

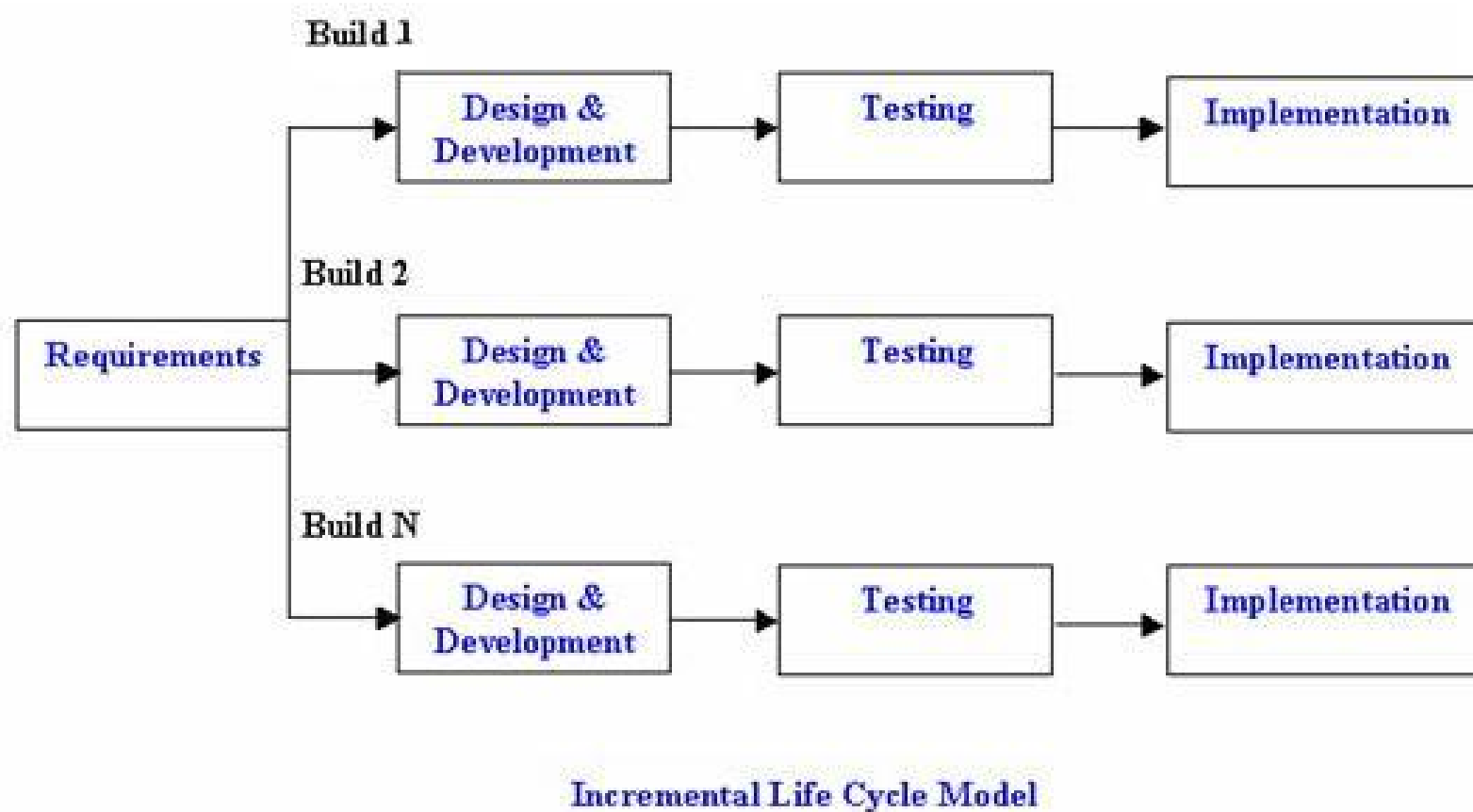
INCREMENTAL MODEL

2. Incremental Model

- **The incremental model is an intuitive approach to the waterfall model**
- **Cycles are divided up into smaller, more easily managed iterations.**
- **Each iteration passes through the requirements, design, implementation and testing phases**

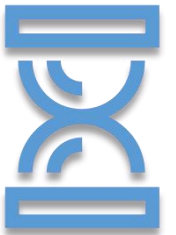
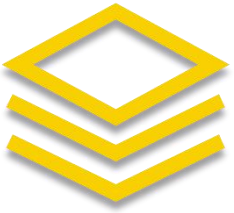


Incremental Model



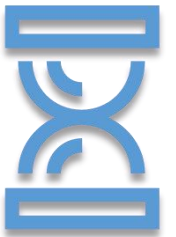
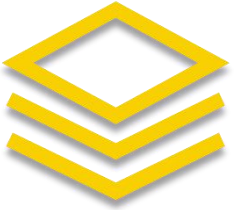
Advantages

- 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



Disadvantages

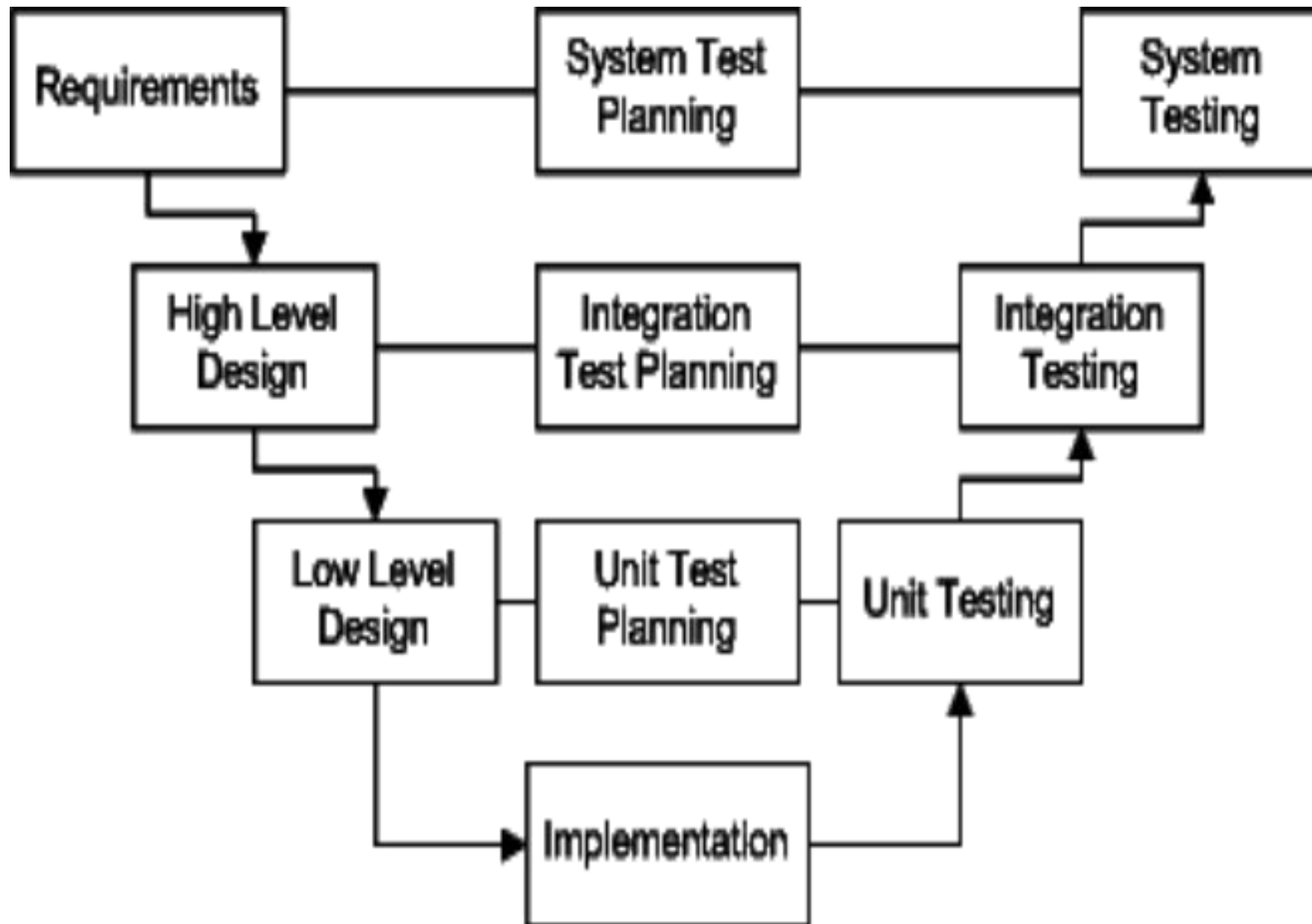
- Needs good planning and design.
- Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
- Total cost is higher than waterfall



V-SHAPED MODEL

3. V-Shaped Model

- The V-Shaped life cycle is a sequential path of execution of processes
- V- model means **Verification** and **Validation** model.



6. V-Shaped Model

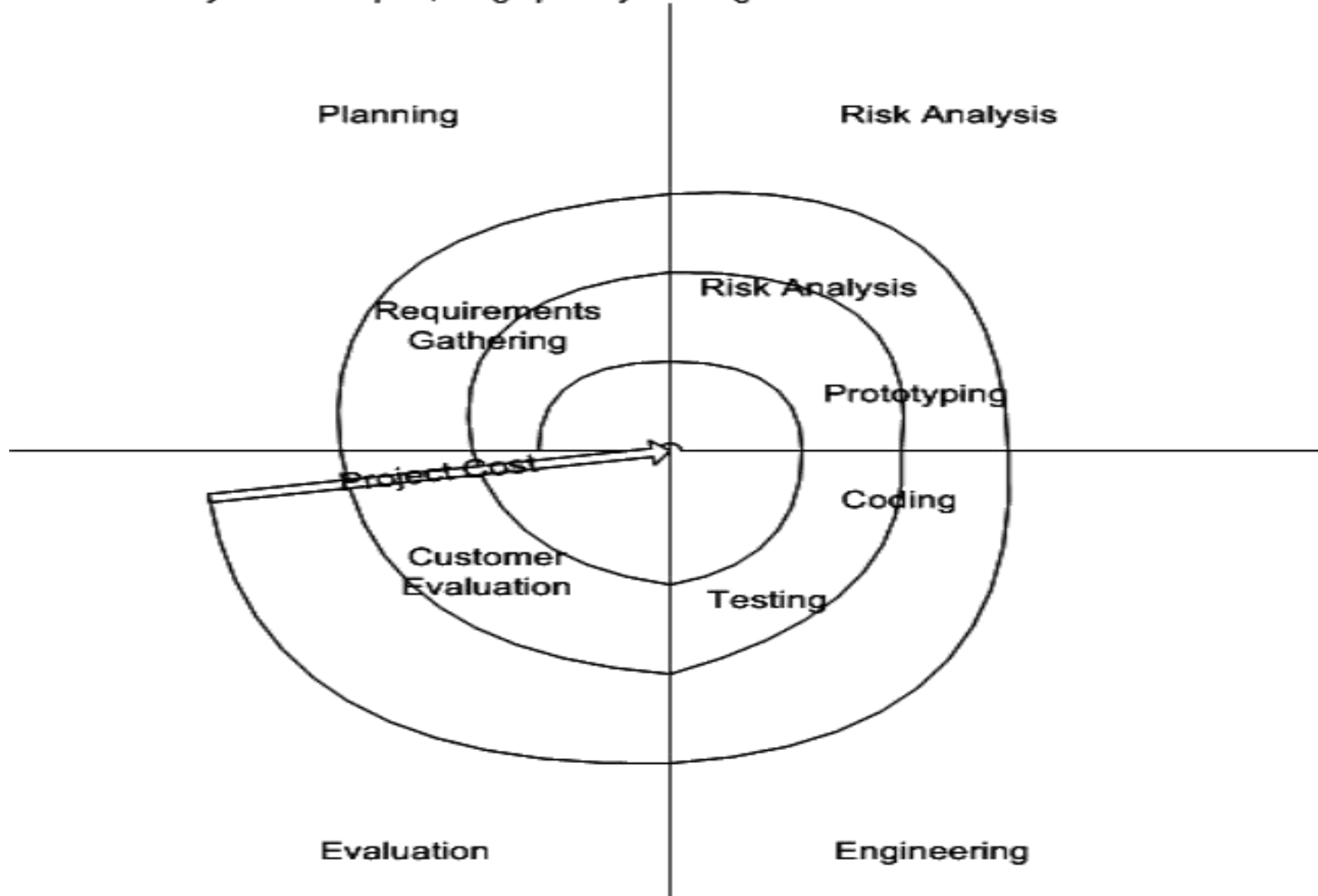
SPIRAL MODEL

4. Boehm's Spiral model(1988)

- **Most generic of the models.**
- **Most life cycle models can be derived as special cases of the spiral model.**
- **Risk management approach to software development.**
- **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**

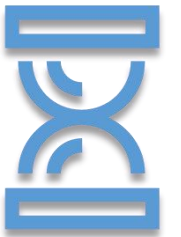
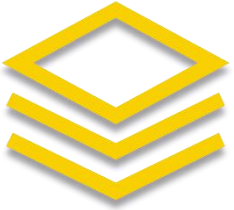
Spiral Model

Sorry about the spiral, I'm graphically challenged.



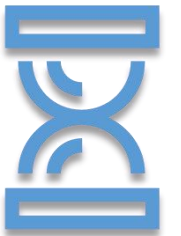
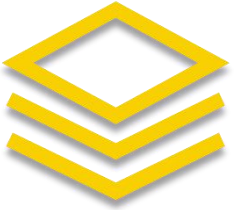
Advantages

- 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](#).



Disadvantages

- 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.




LEAN MODEL

5. Lean

- The seven Lean principles are:
 - eliminate waste,
 - amplify learning,
 - decide as late possible,
 - deliver as fast as possible,
 - empower the team,
 - build integrity in,
 - and see the whole.



Lean (contd.)

- 
- The Lean process is about working only on what must be worked on at the time, so there's **no room for multitasking**.
 - Project teams are also focused on finding opportunities to **cut waste** at every turn throughout the SDLC process, from dropping unnecessary meetings to reducing documentation.

AGILE MODEL

7. Agile

- Speed up or bypass one or more life cycle phases
- Usually less formal and reduced scope
- Used for time-critical applications

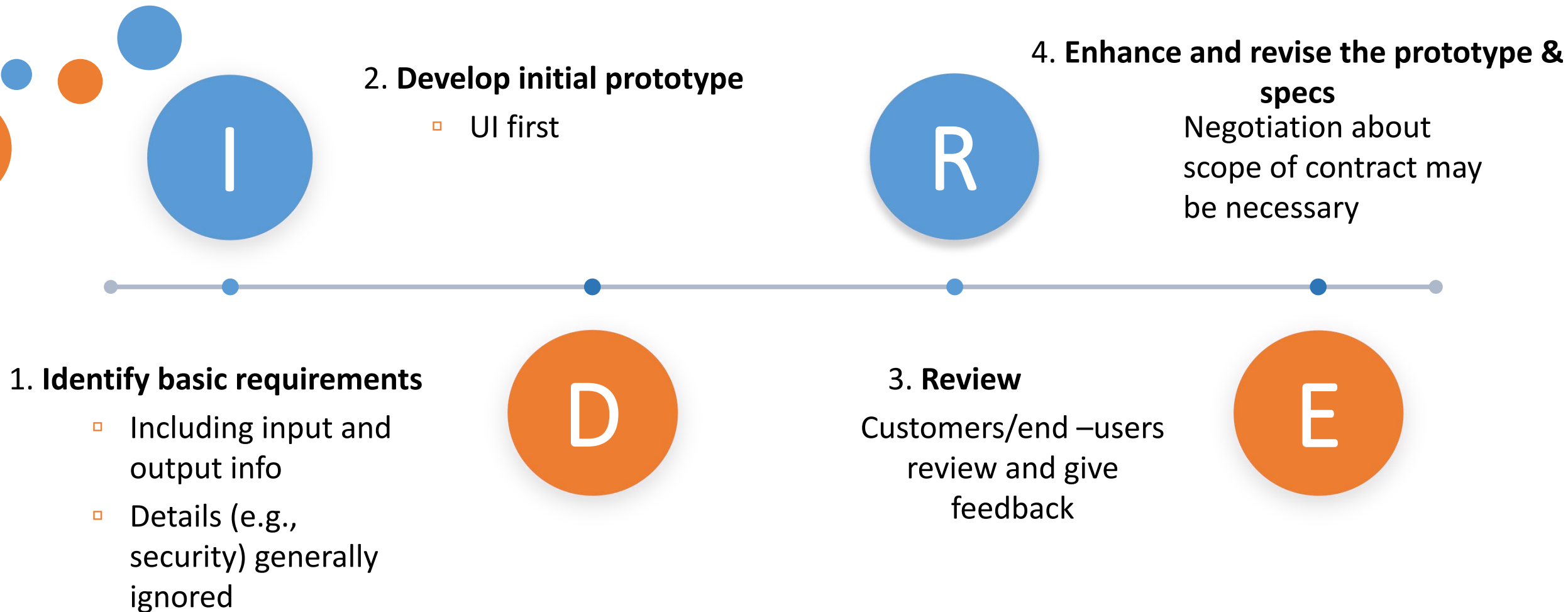


Some Agile Methods

- 
- Rapid Application Development (RAD)
 - Incremental SDLC
 - Scrum
 - Extreme Programming (XP)
 - Adaptive Software Development (ASD)
 - Feature Driven Development (FDD)
 - Crystal Clear
 - Dynamic Software Development Method (DSDM)
 - Rational Unify Process (RUP)

PROTOTYPING MODEL

PROTOTYPING: BASIC STEPS

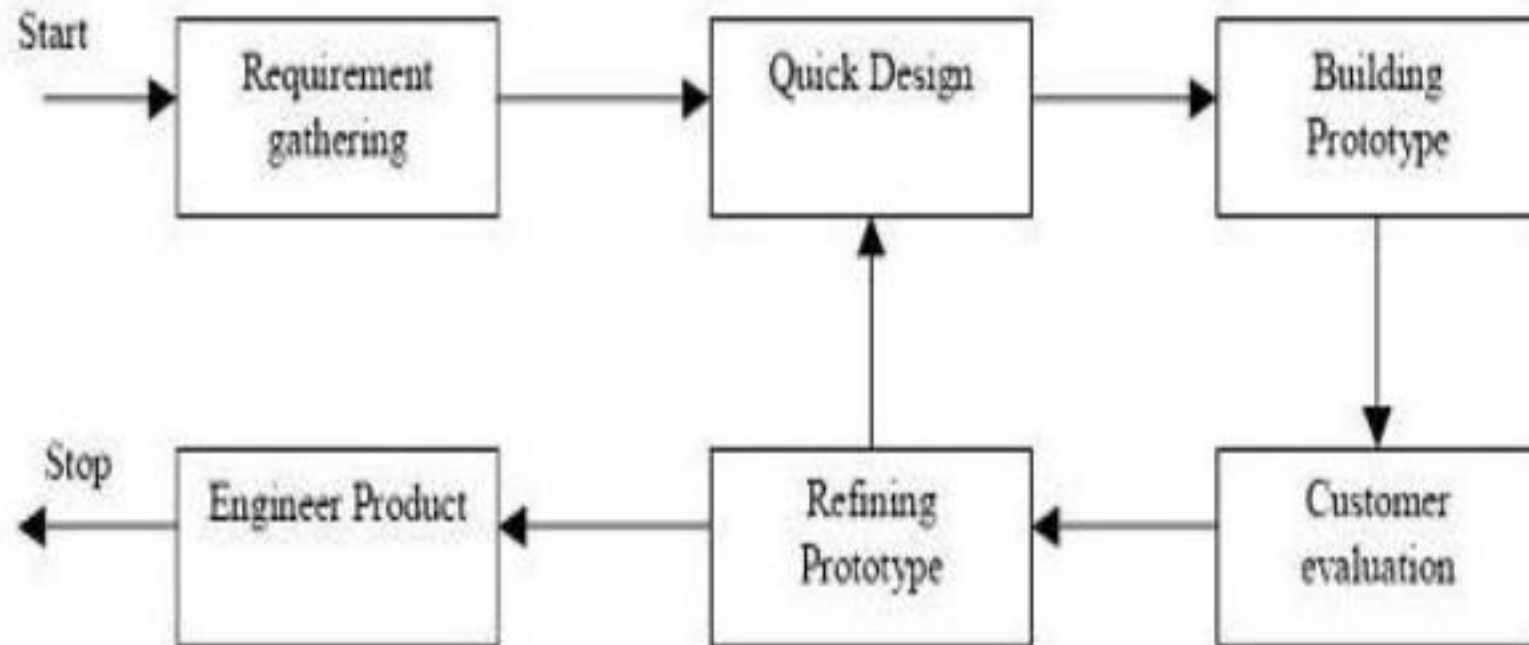


Prototyping

- This prototype is developed based on the currently known requirements
- In the requirement engineering process , a prototype can help with the **elicitation and validation** of the system requirements.



Diagram of Prototype model:



Prototyping Model

Advantages & Disadvantages

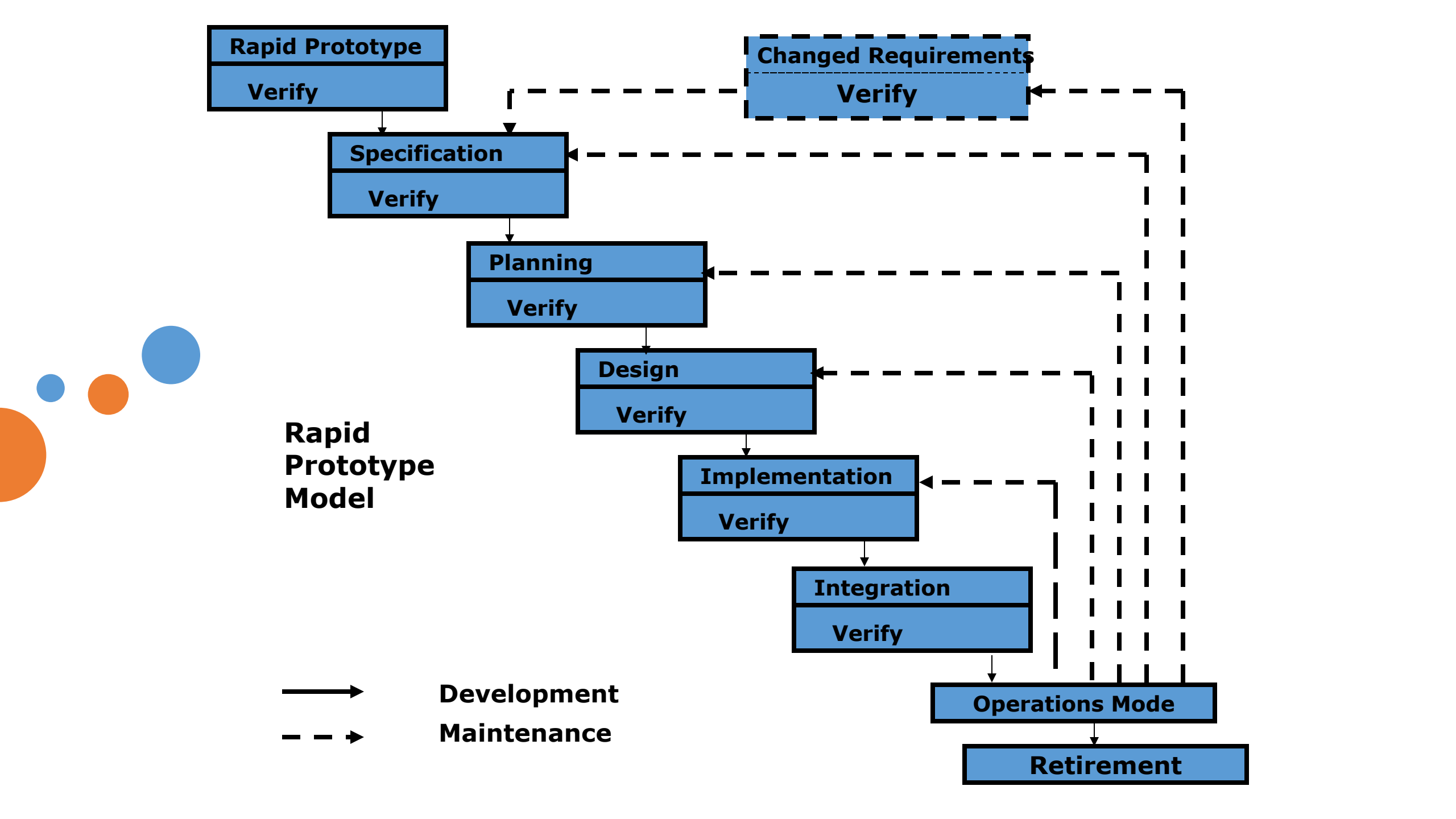
- Users are actively involved in the development
- Errors can be detected much earlier.
- Quicker user feedback is available leading to better solutions.
- Leads to implementing and then repairing way of building systems.
- Prototype model should be used when the desired system needs to have a lot of interaction with the end users.



Rapid prototyping model

- A **rapid prototype** is a working model that is functionally equivalent to a subset of the product





DEVOPS MODEL

DevOps

- Developers and Operations teams work together closely — and sometimes as one team — to accelerate innovation and the deployment of higher-quality and more reliable software products and functionalities.
 - Updates to products are small but frequent.
 - Discipline, continuous feedback and process improvement, and automation of manual development processes are all hallmarks of the DevOps model.

Devops (contd.)

- “DevOps is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.”
 - AWS
- DevOps is not only an approach to planning and executing work, but also a philosophy that demands significant mindset and culture changes in an organization.



END
THANKS

