Chapter 2 – Software Processes

Topics covered

✧ Software process models

✧ Process activities

✧ Coping with change

✧ Process improvement

The software process

- ✧ A structured set of activities required to develop a software system.
- ✧ Many different software processes but all involve:
  - Specification – defining what the system should do;
  - Design and implementation – defining the organization of the system and implementing the system;
  - Validation – checking that it does what the customer wants;
  - Evolution – changing the system in response to changing customer needs.
- ✧ A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

Plan-driven and agile processes

- ✧ Plan-driven processes are processes where all of the process activities are planned in advance and progress is measured against this plan.
- ✧ In agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.
- ✧ In practice, most practical processes include elements of both plan-driven and agile approaches.
- ✧ There are no right or wrong software processes.

Process activities

Process activities

- ✧ Real software processes are inter-leaved sequences of technical, collaborative and managerial activities with the overall goal of specifying, designing, implementing and testing a software system.
- ✧ The four basic process activities of specification, development, validation and evolution are organized differently in different development processes.
- ✧ For example, in the waterfall model, they are organized in sequence, whereas in incremental development they are interleaved.

Software specification

- The process of establishing what services are required and the constraints on the system's operation and development.
- Requirements engineering process
  - Requirements elicitation and analysis
    - What do the system stakeholders require or expect from the system?
  - Requirements specification
    - Defining the requirements in detail
  - Requirements validation
    - Checking the validity of the requirements

Software design and implementation

- ✧ The process of converting the system specification into an executable system.
- ✧ Software design
  - ▪ Design a software structure that realises the specification;
- ✧ Implementation
  - ▪ Translate this structure into an executable program;
- ✧ The activities of design and implementation are closely related and may be inter-leaved.

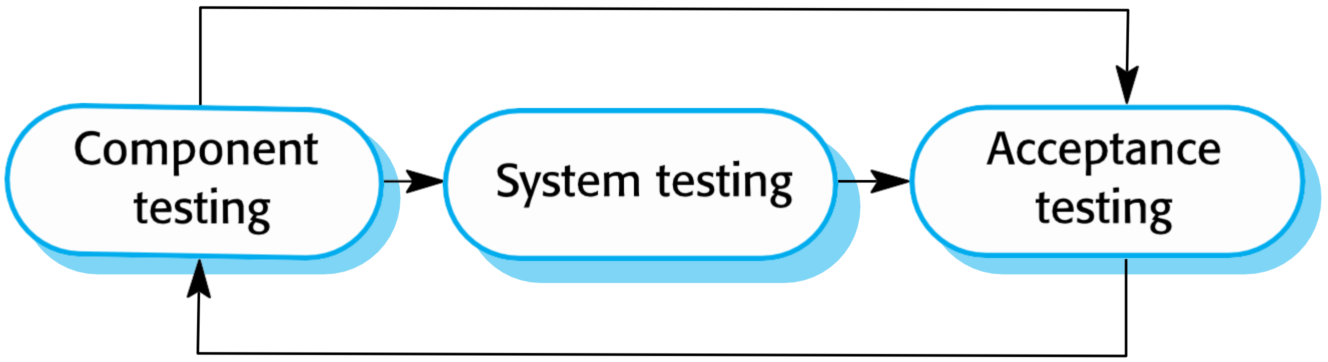
System implementation

- ✧ The software is implemented either by developing a program or programs or by configuring an application system.
- ✧ Design and implementation are interleaved activities for most types of software system.
- ✧ Programming is an individual activity with no standard process.
- ✧ Debugging is the activity of finding program faults and correcting these faults.

Software validation

- Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer.
- Involves checking and review processes and system testing.
- System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system.
- Testing is the most commonly used V & V activity.

Stages of testing



Testing stages

✧ Component testing
- Individual components are tested independently;
- Components may be functions or objects or coherent groupings of these entities.

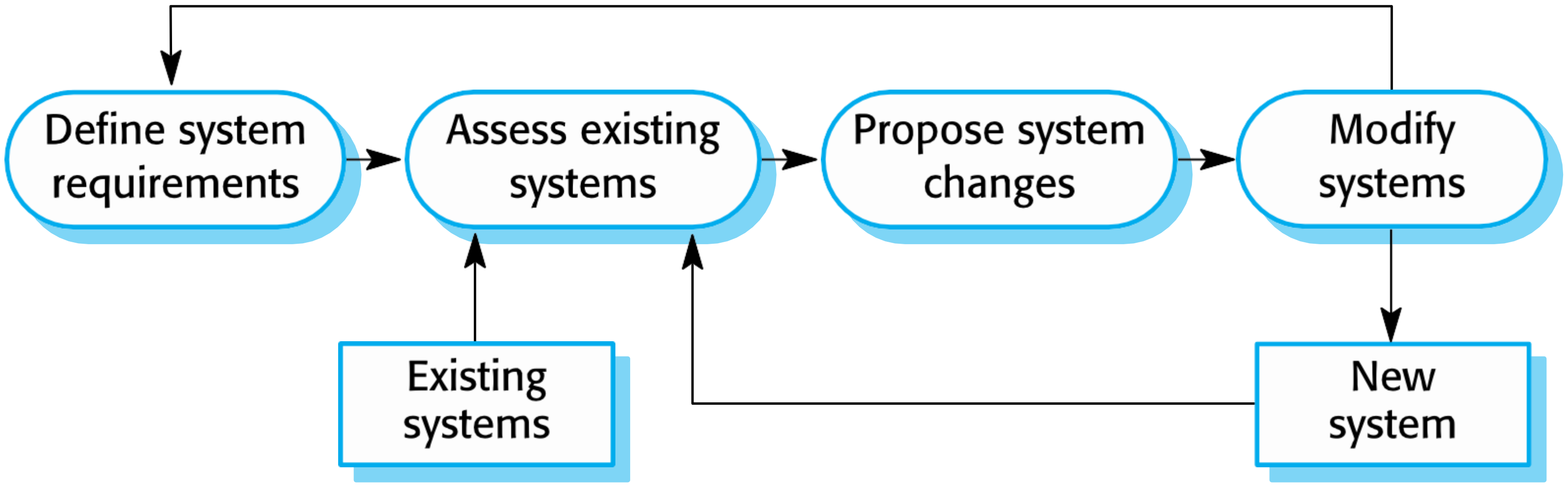
✧ System testing
- Testing of the system as a whole. Testing of emergent properties is particularly important.

✧ Customer testing
- Testing with customer data to check that the system meets the customer’s needs.

Software evolution

- ✧ Software is inherently flexible and can change.
- ✧ As requirements change through changing business circumstances, the software that supports the business must also evolve and change.
- ✧ Although there has been a demarcation between development and evolution (maintenance) this is increasingly irrelevant as fewer and fewer systems are completely new.

System evolution



Software process models

Software process models

- The waterfall model
  - Plan-driven model. Separate and distinct phases of specification and development.
- Incremental development
  - Specification, development and validation are interleaved. May be plan-driven or agile.
- Integration and configuration
  - The system is assembled from existing configurable components. May be plan-driven or agile.
- In practice, most large systems are developed using a process that incorporates elements from all of these models.

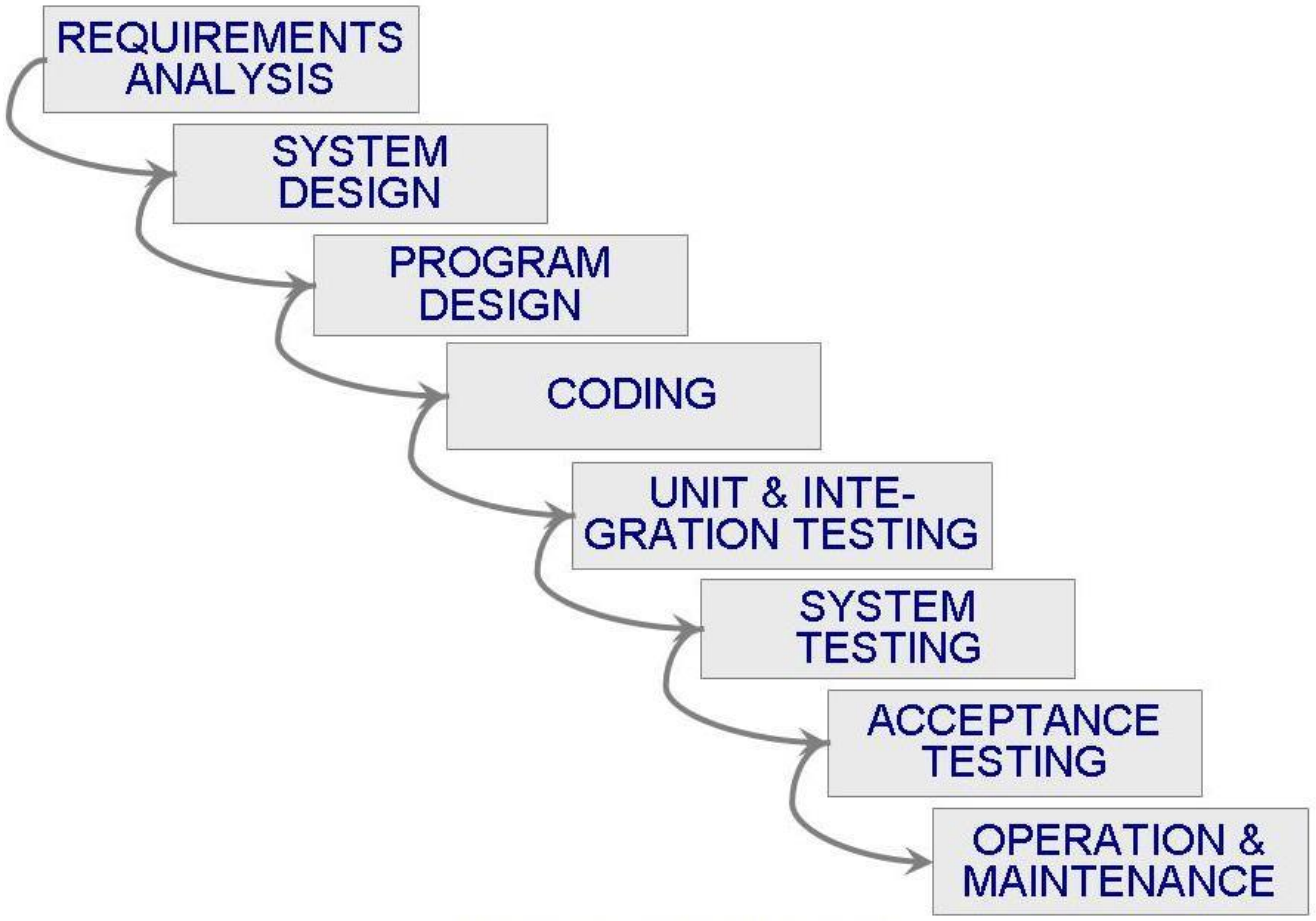
Waterfall

A government contracts a firm to build a bridge.
Requirements are finalized, approved, and legally frozen.
Construction starts only after design completion.
Any change later costs millions.

- Can construction start without a complete design?
- Can the bridge be “iterated” after pouring concrete?

Classical Waterfall was designed for such environments—where change is expensive or dangerous.

Waterfall Model



Waterfall model phases

- There are separate identified phases in the waterfall model:
  - Requirements analysis and definition
  - System and software design
  - Implementation and unit testing
  - Integration and system testing
  - Operation and maintenance
- The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase.

Waterfall model problems

- ✧ Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
  - ▪ Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
  - ▪ Few business systems have stable requirements.
- ✧ The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.
  - ▪ In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

Waterfall

•Originated in **1970s**
•Documentation-heavy, compliance-driven environments
•Low tolerance for failure (nuclear, banking)

Key message:
Classical Waterfall was not “bad design”—
it was rational design for a predictable world.

When to use waterfall model?

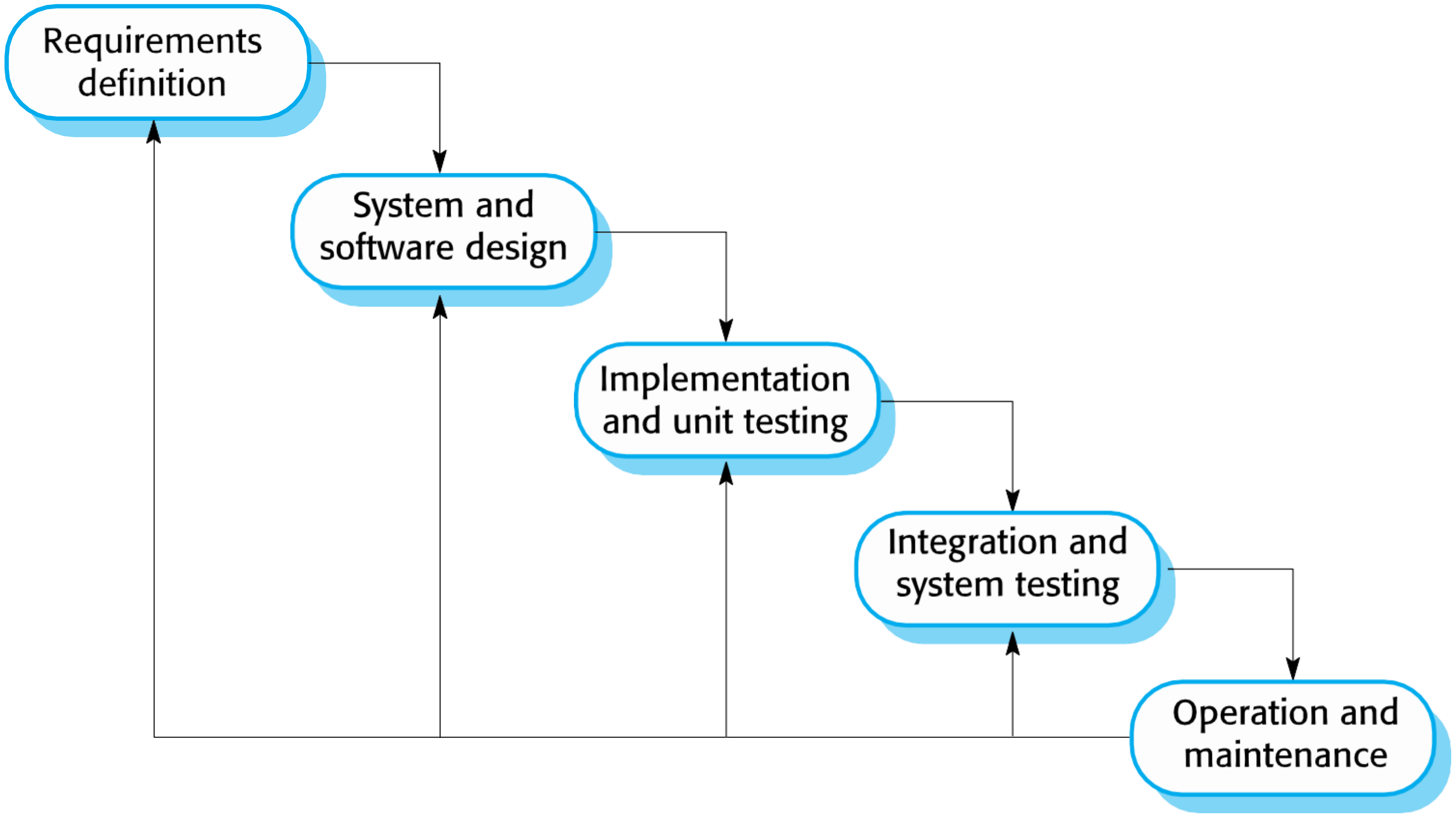
- Clear and Stable Requirements
- Large systems
- Low Risk of Changes
- Well-Defined Technology and Tools
- Documentation-Driven Development

Waterfall

•A payroll or census system delivered exactly as specified
•User needs had changed by deployment time
•Technically correct, practically useless

“What if we kept the structure—but allowed learning between phases?”

The modern waterfall model


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Waterfall with feedback

Changes:

- Overlapping phases
- Early validation and verification
- Limited backtracking
- Continuous documentation updates

Waterfall

Use Classical Waterfall when:
Requirements are stable
Regulatory compliance is critical
Cost of change is extremely high
Use Modern Waterfall when:
Requirements are mostly stable
Feedback is possible but controlled
Large teams need structure

If testing is so important,
why is it always taught at the end of the Waterfall model?

The V-Model was created when engineers realized that testing
should not be an afterthought—it should be planned from day one.

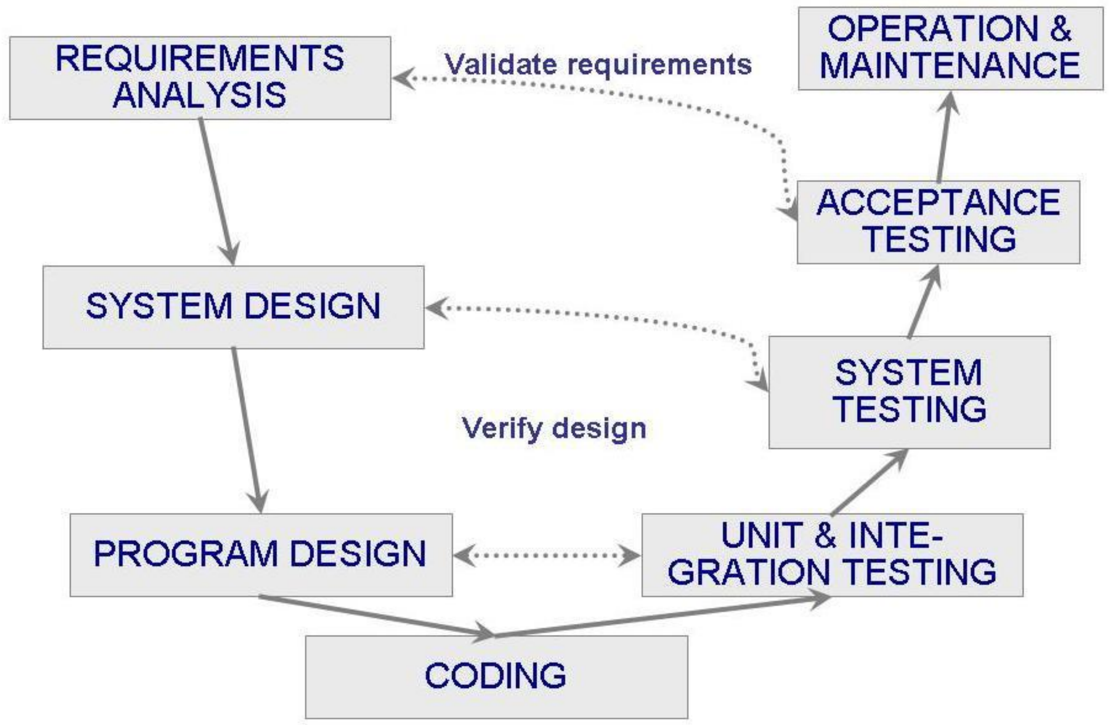
- Waterfall assumes:
  - “First we build everything, then we test.”
- Reality showed:
  - Defects introduced early are **costliest to fix later**.

Every development phase must have a corresponding testing phase

V-Model

A variant 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

V-Model


REQUIREMENTS ANALYSIS
Validate requirements
OPERATION & MAINTENANCE
ACCEPTANCE TESTING
SYSTEM DESIGN
SYSTEM TESTING
Verify design
PROGRAM DESIGN
UNIT & INTE-GRATION TESTING
CODING

Structure of the V-Model

The V-Model is divided into two main branches:

Verification Phases (Left Side of the "V"):

These phases are concerned with the planning, design, and coding aspects of the project.

Validation Phases (Right Side of the "V"):

These phases involve testing and validating the software to ensure that it meets the requirements set during the verification phases.

Phases of the V-Model

1. Requirements Analysis

Objective: Understanding and documenting the functional and non-functional requirements of the system.

Output: Requirements Specification Document.

Corresponding Testing Phase: Acceptance Testing

> **Purpose:** To validate that the system meets the business needs and requirements of the customer.

2. System Design

Objective: Designing the overall system architecture and data flow.

Output: System Design Documents.

Corresponding Testing Phase: System Testing

> **Purpose:** To ensure the system functions correctly as a whole and that it complies with the system specifications.

Phases of the V-Model

Implementation

Objective: Actual coding of the software components as per the Low-Level Design.

Output: Source Code.

Corresponding Testing Phase: Unit Testing

Purpose: To ensure each unit of the code functions correctly and matches its design.

Benefits and Drawbacks if V-Model

Benefits of the V-Model

Early Detection of Defects

Structured Approach

Clear Milestones

Improved Quality

Customer Involvement

Drawbacks of the V-Model

Inflexibility

Not Suitable for Complex or Long-term Projects

High Dependence on Initial Requirements

When to Use the V-Model

The V-Model is most suitable for:
Projects with Clear Requirements: Where requirements are well-understood and unlikely to change.
Safety-Critical Systems: Such as in medical, or automotive industries, where rigorous testing and validation are critical.

Aspect	Waterfall Model	V-Model
Process Flow	Linear and sequential	V-shaped with corresponding testing for each phase
Testing Phase	Testing occurs after the entire development is complete	Testing is integrated at every stage of development
Flexibility	Less flexible; changes are difficult to implement	More flexible due to early defect detection
Risk Management	Higher risk of late defect discovery	Lower risk; defects are identified early
Project Suitability	Suitable for large projects with clear requirements	Suitable for projects requiring rigorous validation

Definitions

- **Iterate**
  - to utter or do repeatedly
  - **iteration *n*. – iterative *adj***
- **Increment *n***
  - 1. amount of increase
  - 2. a becoming greater or larger; increase
    - **incremental *adj***

Incremental Development

- Building and releasing one feature at a time depending on priorities defined by the customer
  - You have all the requirements
  - you design the complete product first. You only leave out details that you can safely decide later.
  - Then you slice it up into chunks and build each separately.
  - When you finish a chunk, aka module, you integrate it with previously completed parts, so they work as a whole.

Example: Ecommerce website

- ❑Consider a team building an ecommerce website using incremental development. The final target product has search, product information, a shopping basket, checkout, favourites, and customer reviews.
- ❑For the first released increment, the team builds the basic functionality to buy a product. It includes search, product information, adding products to a shopping basket and checkout. This first slice would only be released once it’s complete.
- ❑The second released increment builds on that basic functionality, and would add another capability such as favourites. The would be released when the favourites functionality is complete.
- ❑The third released increment adds customer reviews once that is complete, and so on.

Iterative Development

Iterative development is a lot like inventing: discovering what and how you need as you go.
Build the overall solution/product and then refine some of the areas that require improvement

You start with a fair idea of what you want the product to accomplish, and you use a process of successive approximation to design and build it.

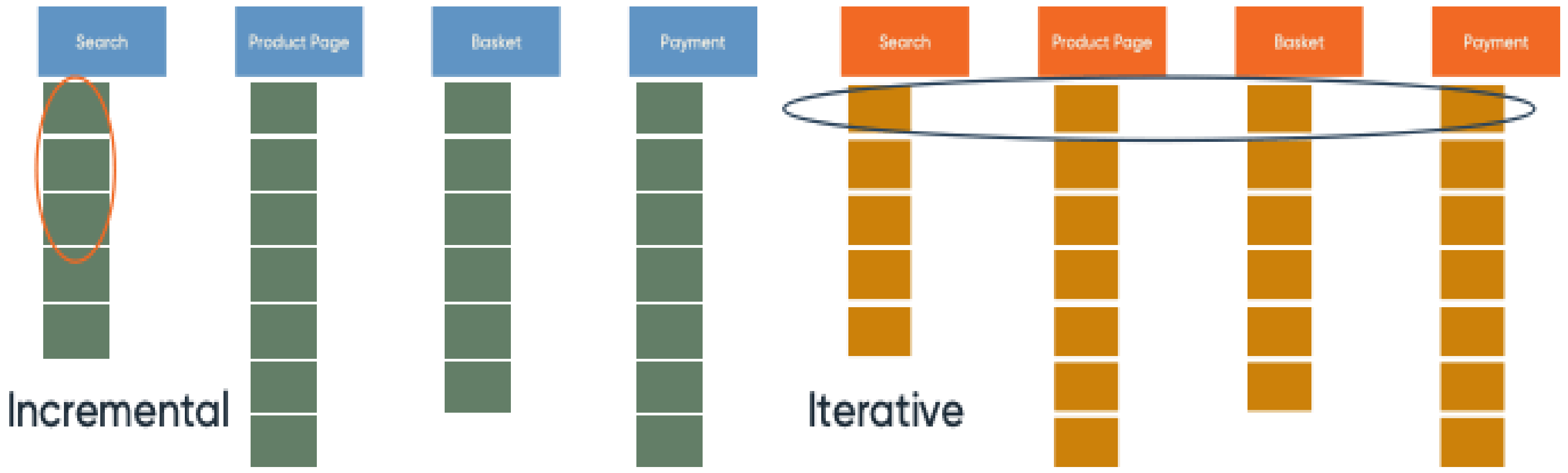
You begin by designing, building, and testing the tiniest version of what you have in mind. When you’re happy, you show it and collect feedback from everybody with a stake in the product.

If what you created was well received, you keep it and expand on it in the next iteration. If what you created got the thumbs down, you discard it and go back to the drawing board

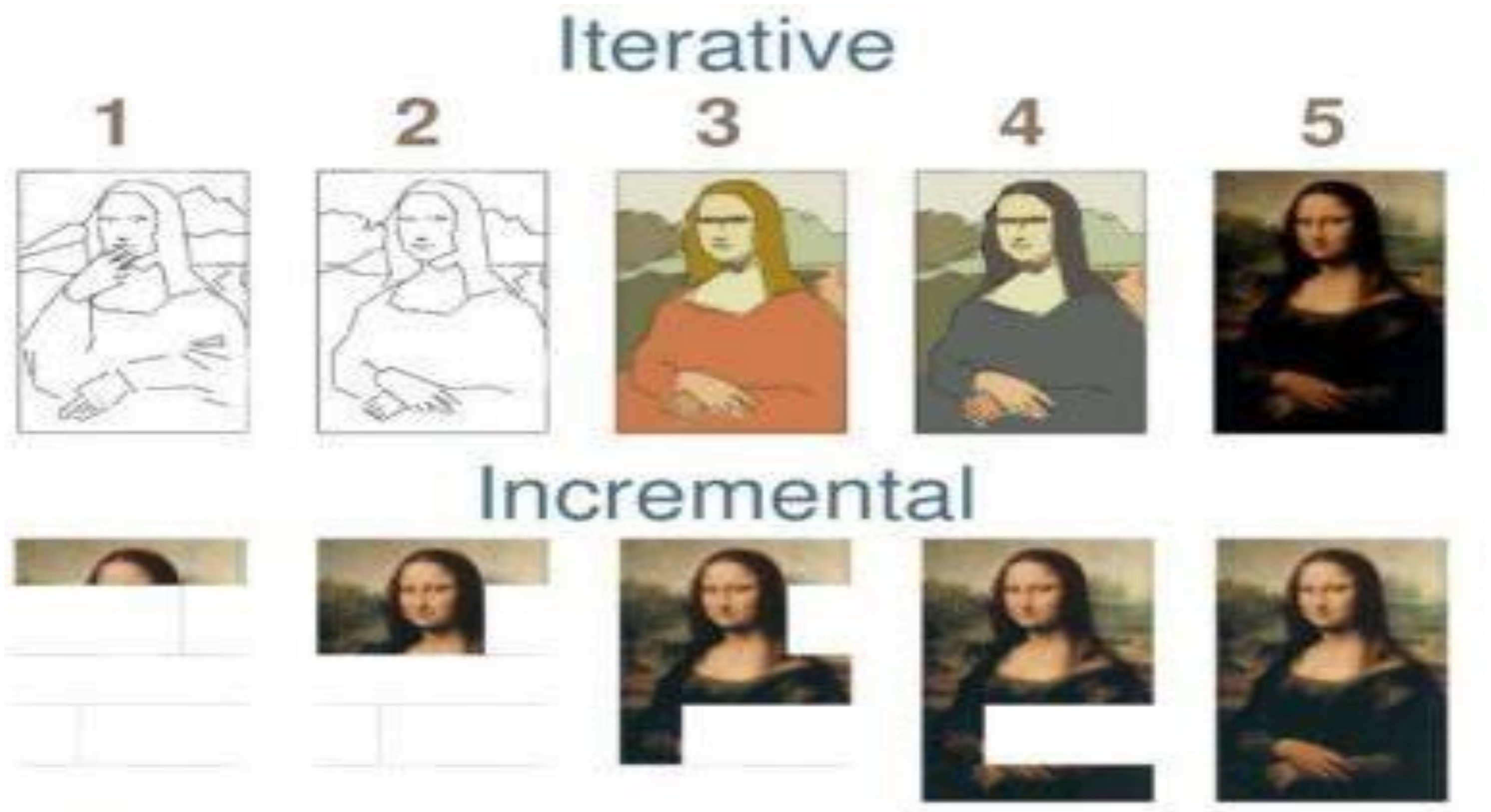
Example: Ecommerce website

- ❑Assume a team building the same ecommerce website using an iterative process.
- ❑The first release has a really stripped back version of all the required functionality; namely search, product information, a shopping basket, checkout, favourites, and customer reviews.
- ❑For the second iterative release, the team would improve some of the existing basic functionality, taking into account feedback from stakeholders or customer, or other inputs such as analytics.
- ❑On every subsequent iterative release, new ideas and requirements are added or low value/usage areas may be removed.

Example: Ecommerce website



Example: Artist painting a picture



Coping with change

Coping with change

- ✧ Change is inevitable in all large software projects.
  - Business changes lead to new and changed system requirements
  - New technologies open up new possibilities for improving implementations
  - Changing platforms require application changes
- ✧ Change leads to rework so the costs of change include both rework (e.g. re-analysing requirements) as well as the costs of implementing new functionality

Coping with changing requirements

- ✧ System prototyping, where a version of the system or part of the system is developed quickly to check the customer's requirements and the feasibility of design decisions. This approach supports change anticipation.
- ✧ Incremental delivery, where system increments are delivered to the customer for comment and experimentation. This supports both change avoidance and change tolerance.

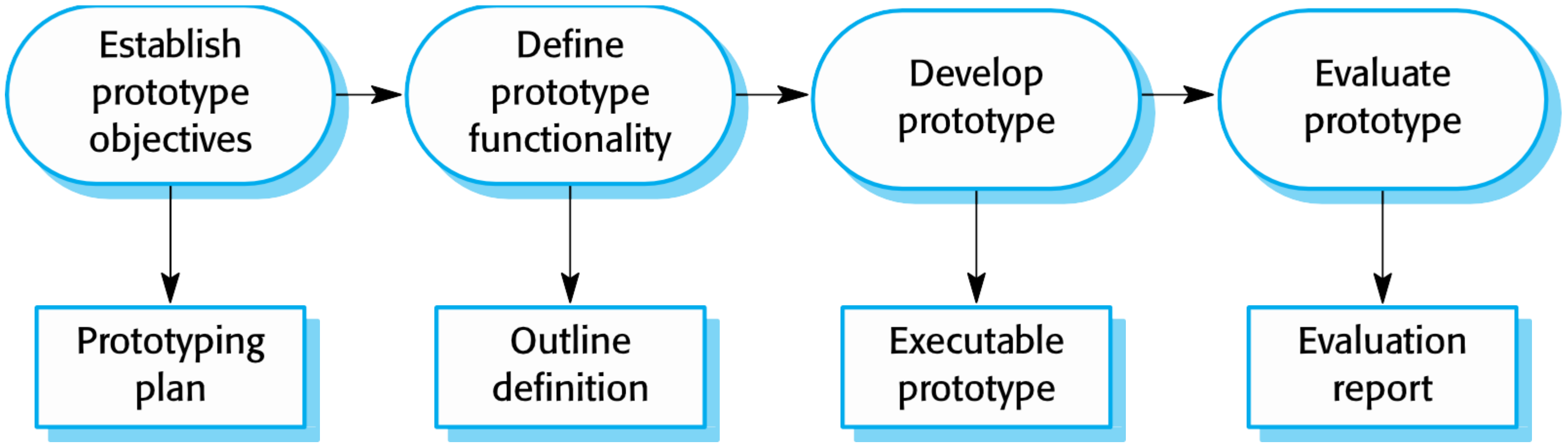
Software prototyping

- ✧ A prototype is an initial version of a system used to demonstrate concepts and try out design options.
- ✧ A prototype can be used in:
  - The requirements engineering process to help with requirements elicitation and validation;
  - In design processes to explore options and develop a UI design;
  - In the testing process to run back-to-back tests.

Benefits of prototyping

- ✧ Improved system usability.
- ✧ A closer match to users' real needs.
- ✧ Improved design quality.
- ✧ Improved maintainability.
- ✧ Reduced development effort.

The process of prototype development

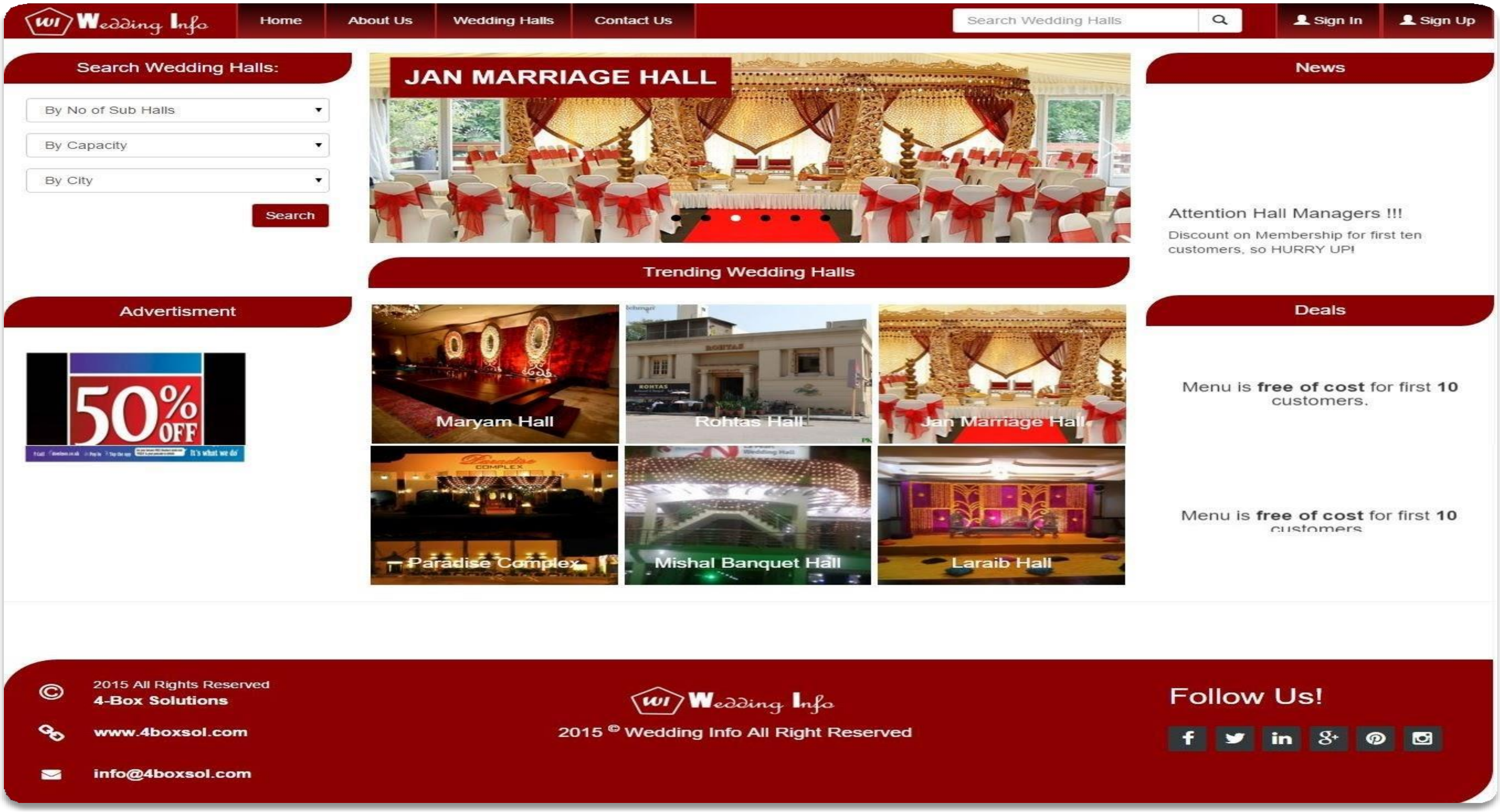


Prototype development

- ✧ May be based on rapid prototyping languages or tools
- ✧ May involve leaving out functionality
  - Prototype should focus on areas of the product that are not well-understood;
  - Error checking and recovery may not be included in the prototype;
  - Focus on functional rather than non-functional requirements such as reliability and security

Throw-away prototypes

- ✧ Prototypes should be discarded after development as they are not a good basis for a production system:
    - It may be impossible to tune the system to meet non-functional requirements;
    - Prototypes are normally undocumented;
    - The prototype structure is usually degraded through rapid change;
    - The prototype probably will not meet normal organisational quality standards.



Attention Hall Managers !!!

Discount on Membership for first ten customers, so HURRY UP!

Menu is **free of cost** for first **10** customers.

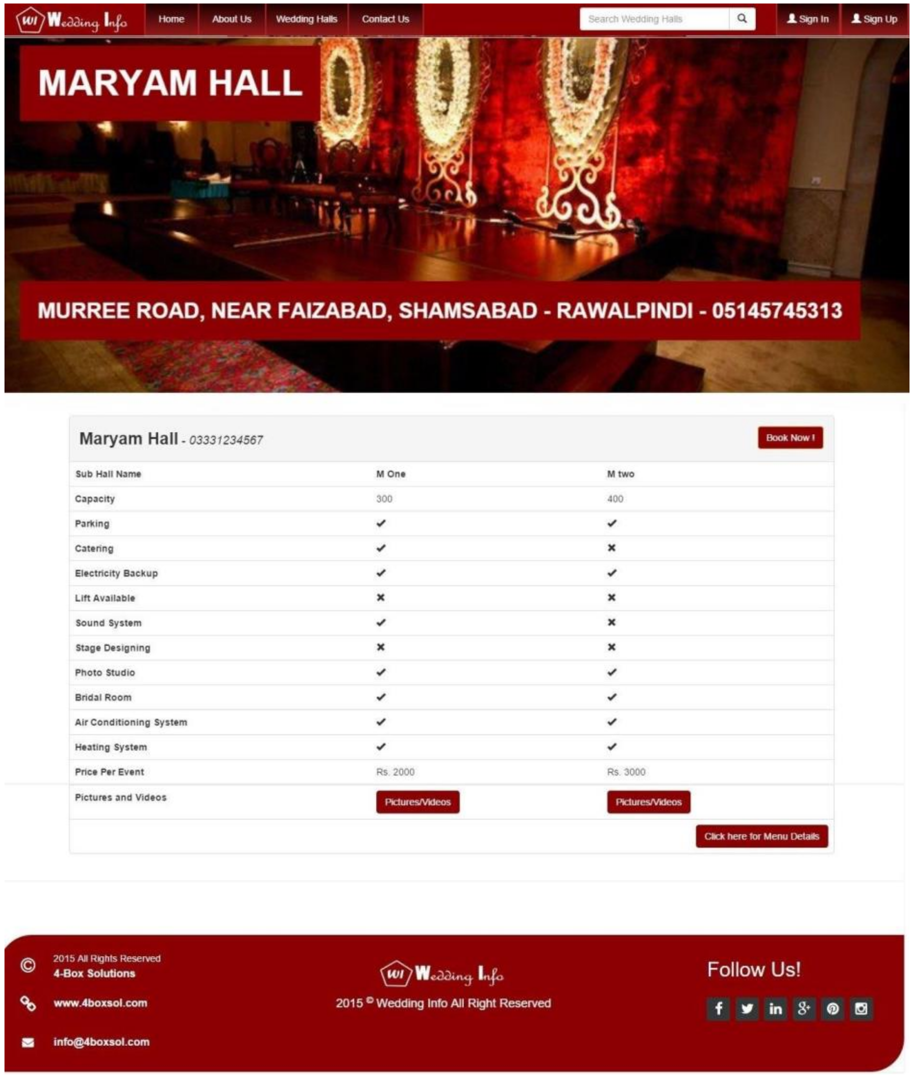
Menu is **free of cost** for first **10** customers

2015 All Rights Reserved
4-Box Solutions

www.4boxsol.com

info@4boxsol.com

2015 © Wedding Info All Right Reserved



MARYAM HALL

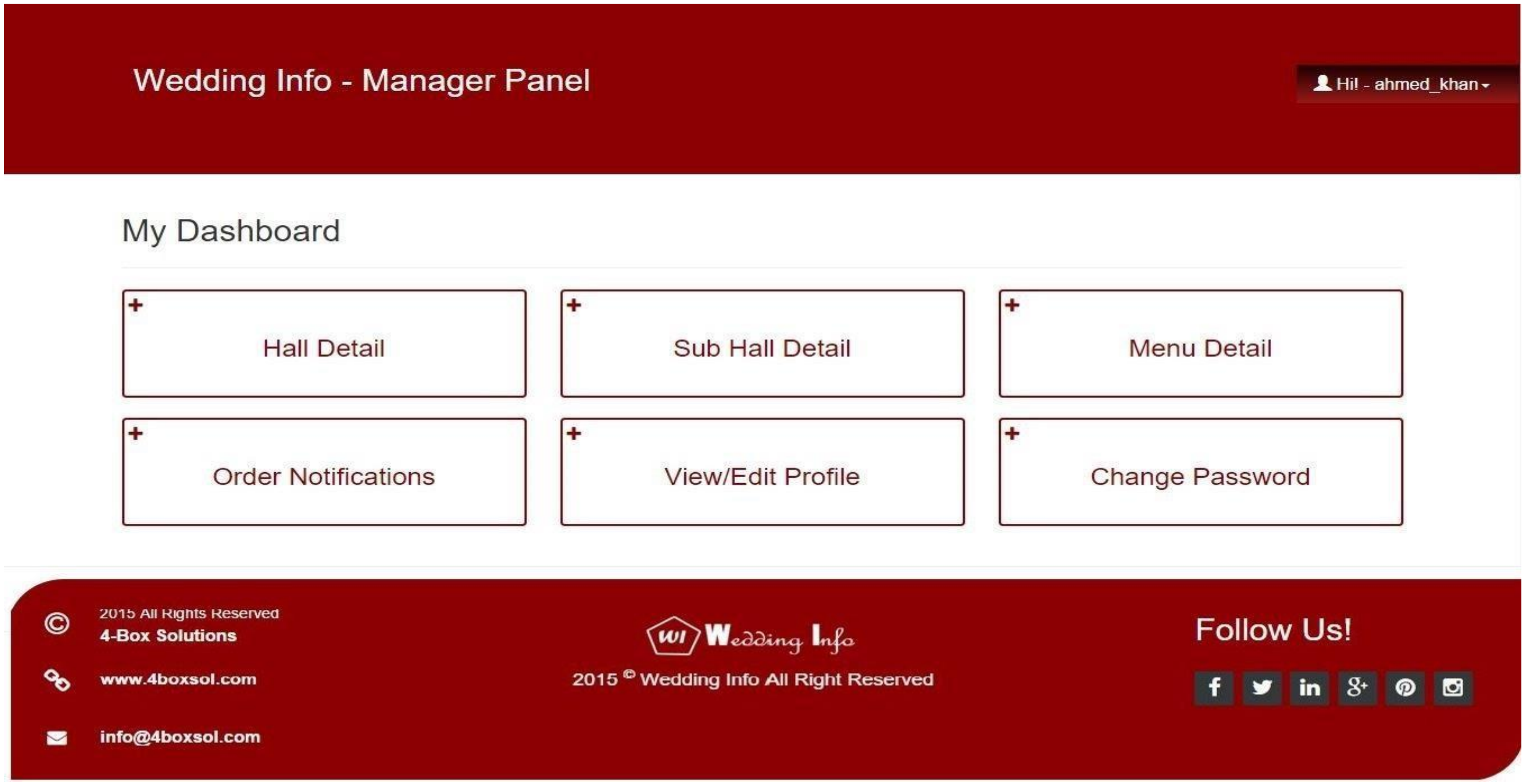
MURREE ROAD, NEAR FAIZABAD, SHAMSABAD - RAWALPINDI - 05145745313

Maryam Hall - *03331234567*

Sub Hall Name	M One	M two
Capacity	300	400
Parking	✓	✓
Catering	✓	✗
Electricity Backup	✓	✓
Lift Available	✗	✗
Sound System	✓	✗
Stage Designing	✗	✗
Photo Studio	✓	✓
Bridal Room	✓	✓
Air Conditioning System	✓	✓
Heating System	✓	✓
Price Per Event	Rs. 2000	Rs. 3000
Pictures and Videos	Pictures/Videos	Pictures/Videos

2015 All Rights Reserved
4-Box Solutions
www.4boxsol.com
info@4boxsol.com

2015 © Wedding Info All Right Reserved



Wedding Info - Manager Panel
Hi! - ahmed_khan
My Dashboard
Hall Detail
Sub Hall Detail
Menu Detail
Order Notifications
View/Edit Profile
Change Password
2015 All Rights Reserved
4-Box Solutions
www.4boxsol.com
info@4boxsol.com
WI Wedding Info
2015 © Wedding Info All Right Reserved
Follow Us!

Tools

Figma

Boehm's Spiral Model

Process is represented as a spiral rather than as a sequence of activities with backtracking.

Each loop in the spiral represents a phase in the process .

No fixed phases such as specification or design loops in the spiral are chosen depending on what is required.

Risks are explicitly assessed and resolved throughout the process.

Boehm’s Spiral Model

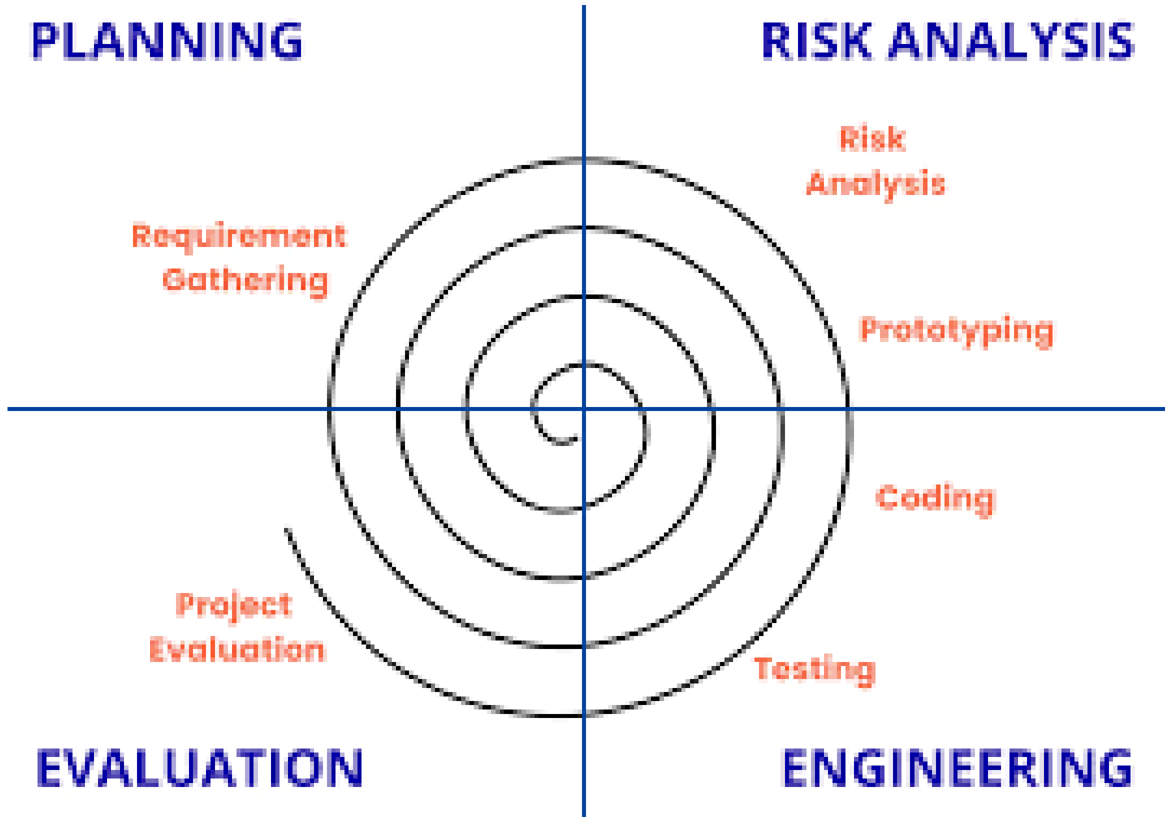
Process is represented as a spiral rather than as a sequence of activities with backtracking.

Each loop in the spiral represents a phase in the process .

No fixed phases such as specification or design loops in the spiral are chosen depending on what is required.

Risks are explicitly assessed and resolved throughout the process.

Boehm's Spiral Model



Spiral Model Usage

Spiral model has been very influential in helping people think about iteration in software processes and introducing the risk driven approach to development.

In practice, however, the model is rarely used:

- Can be a costly model to use
- Risk analysis requires highly specific expertise
- Doesn’t work well for smaller projects.

When to use spiral model

The following pointers explain the typical uses of a Spiral Model –
For medium to high-risk projects.
Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
Customer is not sure of their requirements which is usually the case.
Requirements are complex and need evaluation to get clarity.
New product line which should be released in phases to get enough customer feedback.
Significant changes are expected in the product during the development cycle.

Advantages & Disadvantages

Benefits	Drawbacks
Allows the use prototyping	Management is more complex
Client can see the system early on	Should not be used for small projects
Changes in requirements can be made easily	This method requires a lot of documentation

Aspect	**Waterfall Model**	**V-Model**	**Spiral Model**
When to Use	- Clear, stable requirements	- Projects requiring rigorous testing and validation	- Complex, high-risk projects with evolving requirements
:	- Large projects	- Safety-critical systems (e.g., medical, aviation)	- Projects requiring frequent revisions and risk analysis
	- Well-defined technology and tools	- Where defect detection at every stage is crucial	- Large-scale projects with uncertain requirements
	- Minimal changes expected after the project starts	- When thorough documentation and traceability are needed	- Projects where flexibility and iterative development are essential
Examples	- Govt Payroll management systems	- Medical device software	- Aerospace or defense systems