

Software Engineering

Prepared by Dr. Yasser Abdelhamid

IT 382
Software Engineering

Lecture 4



LECTURE OUTLINE

- ◉ **Coping with change.**
- ◉ **Sources of change**
- ◉ **Cost of change**
- ◉ **How to reduce cost of change**
- ◉ **Prototyping**
- ◉ **Incremental development and delivery**

Change is **inevitable** in all large software projects.

SOURCES OF CHANGE

- **Business requirements** changes
- **New technologies** emerge which open up new possibilities for improving implementations
- **Changing platforms** require application changes

COST OF CHANGE

- **costs of change**
 - cost of **rework** (e.g., re-analyzing requirements)
 - costs of **implementing** new functionality

REDUCING THE COSTS OF REWORK

- Two ways to reduce cost of change
 - Change avoidance
 - Change tolerance

CHANGE AVOIDANCE

- **Change avoidance**, can be accomplished by **anticipating** possible changes.
 - For example, a **prototype** system may be developed to show some key features of the system to customers.

CHANGE TOLERANCE

- **Change tolerance** can be accomplished by making changes accommodated at relatively **low cost**.
 - Using **incremental development** approach
 - Proposed changes may be implemented in increments that have not yet been developed.

SOFTWARE PROTOTYPING

- **What is a prototype?**

- A prototype is an **initial version** of a system.
- It is used to:
 - **Demonstrate concepts.**
 - **Try out design options.**

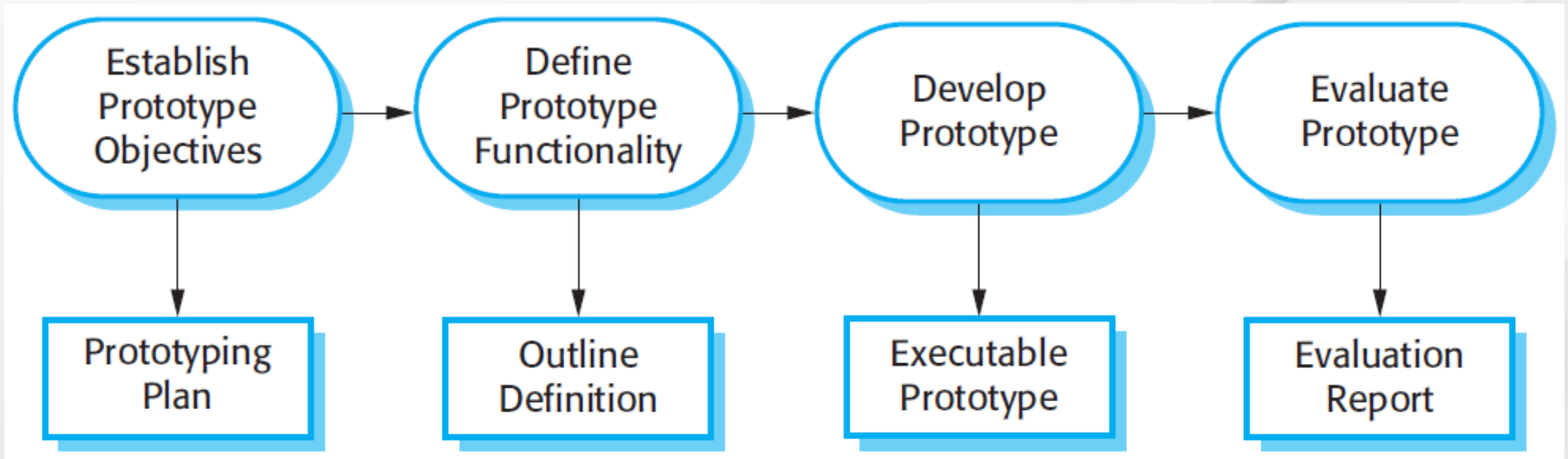
SOFTWARE PROTOTYPING

- How a prototype can be used?
 - Prototyping can be used for:
 - The **requirements engineering** process to help with **requirements elicitation and validation**;
 - In **design processes** to **explore options** and develop a **User Interface 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 FEATURES

A prototype may involve **leaving out
functionality**

PROTOTYPE DEVELOPMENT FEATURES

Prototype should **focus** on areas of the product that are **not well-understood**

**Error checking and recovery may
not be included in the prototype**

PROTOTYPE DEVELOPMENT FEATURES

Focus on **functional** rather than non-functional requirements such as reliability and security

QUESTION?

Should we **use the prototype as a part of the real product**, or should we **throw it away**, and start building the system from the beginning

?

THROW-AWAY PROTOTYPES

Prototypes **should be discarded** after development as they are not a good basis for a production system.

WHY PROTOTYPES SHOULD BE DISCARDED?

(1)

It may be impossible to tune the system to meet non-functional requirements;

WHY PROTOTYPES SHOULD BE DISCARDED?

(2)

Prototypes are normally
undocumented;

WHY PROTOTYPES SHOULD BE DISCARDED?

(3)

The prototype **structure is usually degraded** through rapid change;

WHY PROTOTYPES SHOULD BE DISCARDED?

(4)

The prototype probably will not meet normal
organizational quality standards.

INCREMENTAL DELIVERY

Rather than deliver the system as a single delivery, the development and delivery is broken down into **increments** with each increment delivering part of the required functionality.

INCREMENTAL DELIVERY

User requirements are **prioritised** and the highest priority requirements are included in **early increments**.

INCREMENTAL DELIVERY

Once the development of an increment is started, the **requirements are frozen** though requirements for later increments can continue to evolve.

INCREMENTAL DEVELOPMENT AND DELIVERY

- **Incremental development**

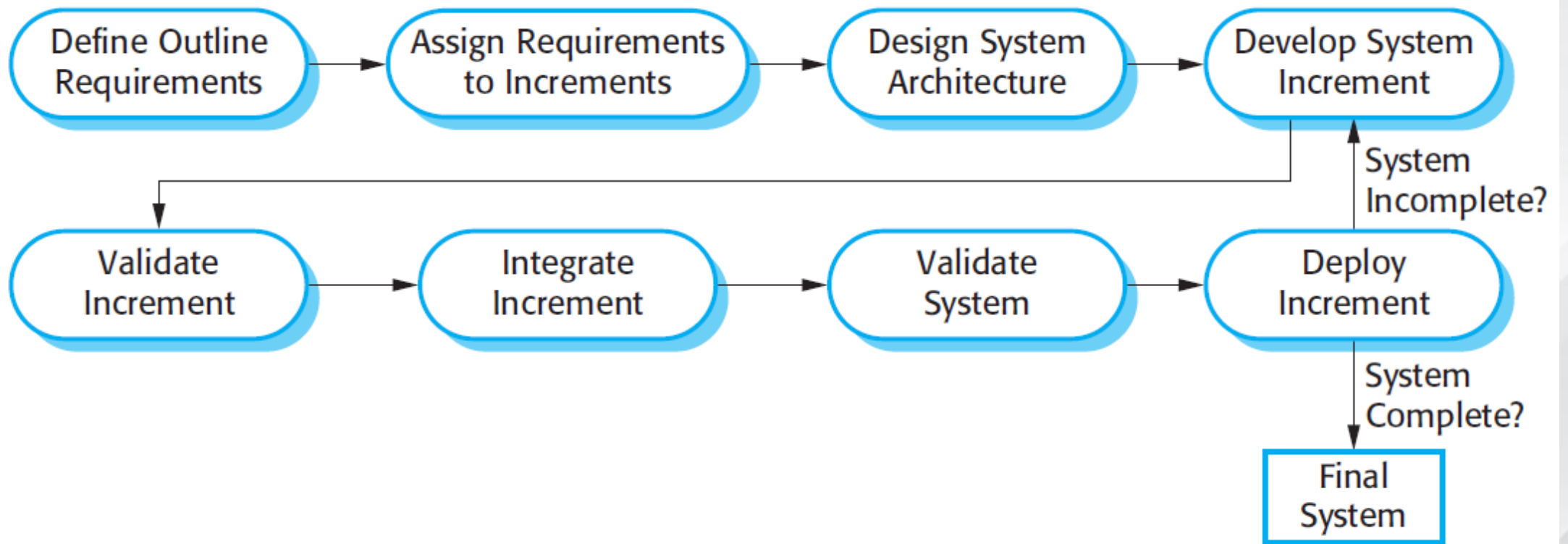
- Develop the system in increments and **evaluate each increment** before proceeding to the development of the next increment;
- Normal approach used in **agile** methods;
- Evaluation done by **user/customer**.

INCREMENTAL DEVELOPMENT AND DELIVERY

- **Incremental delivery**

- Deploy an increment for use by **end-users**;
- More **realistic evaluation** about practical use of software;
- But difficult to implement for **replacement systems** as **increments have less functionality** than the system being replaced.

INCREMENTAL DELIVERY



INCREMENTAL DELIVERY ADVANTAGES

(1)

Customer value can be **delivered** with each increment so system functionality is available **earlier**.

INCREMENTAL DELIVERY ADVANTAGES

(2)

Early **increments** act as a **prototype** to help **elicit** requirements for later increments.

INCREMENTAL DELIVERY ADVANTAGES

(3)

Lower risk of overall project **failure.**

INCREMENTAL DELIVERY ADVANTAGES

(4)

The **highest priority** system **services** tend to receive the **most testing**.

INCREMENTAL DELIVERY PROBLEMS

(1)

Most systems require a **set of basic facilities** that are used by different parts of the system.

As requirements are not defined in detail until an increment is to be implemented, **it can be hard to identify common facilities** that are needed by all increments.

INCREMENTAL DELIVERY PROBLEMS

(2)

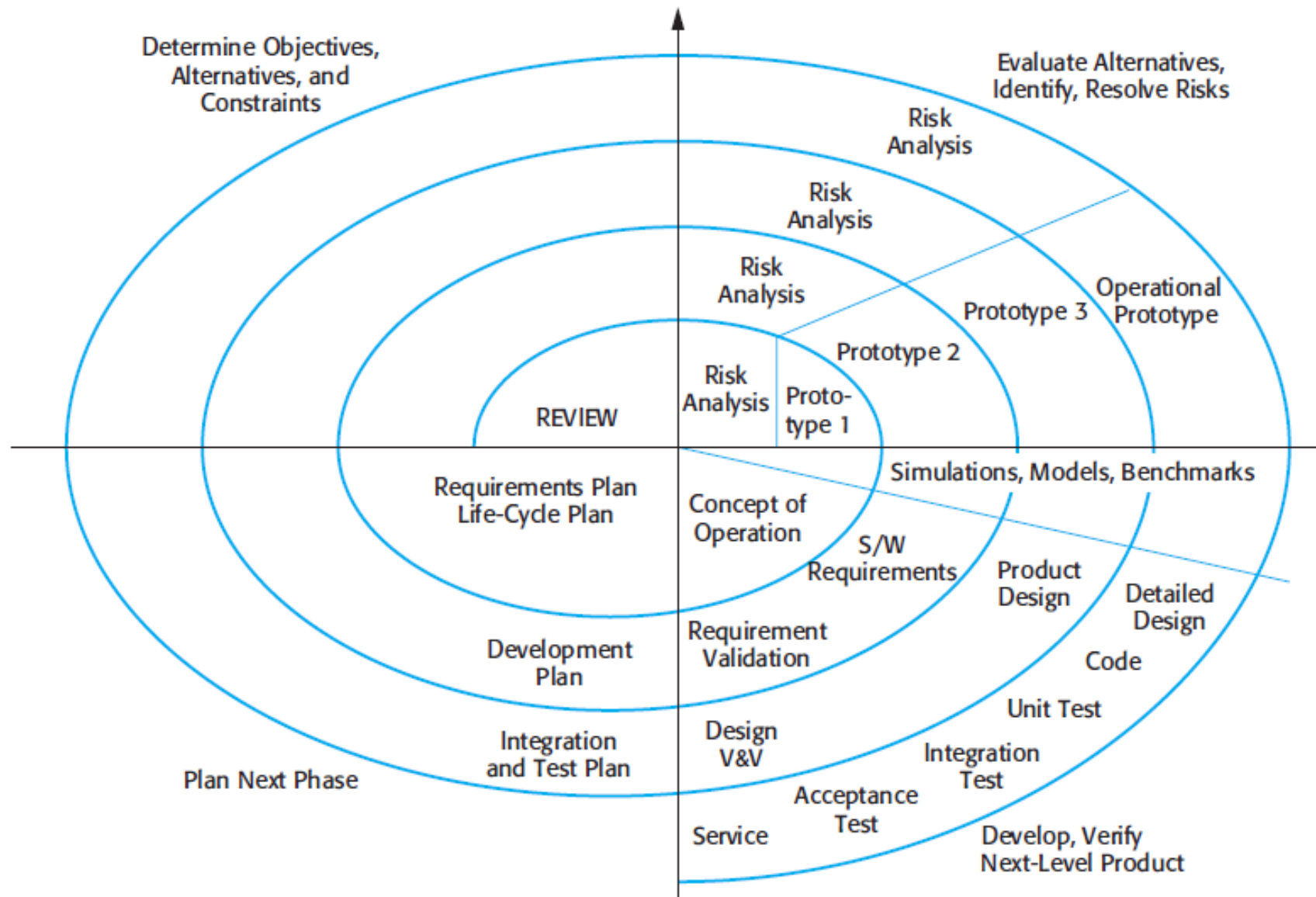
The essence of iterative processes is that the **specification is developed in conjunction with the software.**

This conflicts with the **procurement model**, where the complete system specification is part of the system **development contract.**

BOEHM'S SPIRAL MODEL

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

Objective setting

- Specific objectives for the phase are identified.

SPIRAL MODEL SECTORS

Risk assessment and reduction

- **Risks** are **identified** and assessed and
- Activities put in place to **reduce** the key risks.

SPIRAL MODEL SECTORS

Development and Validation

- A development model for the system is chosen which can be **any of the generic models.**

SPIRAL MODEL SECTORS

Planning

- The **project** is **reviewed** and the **next phase** of the spiral is **planned**.

EXAMPLES OF RISKS

Risk	Affects	Description
Staff turnover	Project	Experienced staff will leave the project before it is finished.
Management change	Project	There will be a change of organizational management with different priorities.
Hardware unavailability	Project	Hardware that is essential for the project will not be delivered on schedule.
Requirements change	Project and product	There will be a larger number of changes to the requirements than anticipated.
Specification delays	Project and product	Specifications of essential interfaces are not available on schedule.
Size underestimate	Project and product	The size of the system has been underestimated.
CASE tool underperformance	Product	CASE tools, which support the project, do not perform as anticipated.
Technology change	Business	The underlying technology on which the system is built is superseded by new technology.
Product competition	Business	A competitive product is marketed before the system is completed.

SPIRAL MODEL USAGE

- Spiral model has been very influential in helping people think about **iteration in software processes**.
- introducing the **risk-driven** approach to development.
- But the model is rarely used as published for practical software development.



SUMMARY

- ◉ **Coping with change.**
- ◉ **Sources of change**
- ◉ **Cost of change**
- ◉ **How to reduce cost of change**
- ◉ **Prototyping**
- ◉ **Incremental development and delivery**
- ◉ **Spiral model**



Thank You

