Lecture 4: Software Process - II

Software Engineering (CSM 31212)

MM. Mohamed Mufassirin

Lecturer in Computer Science

Dept. of Computer Science

FAS/SEUSL

Software Process Models

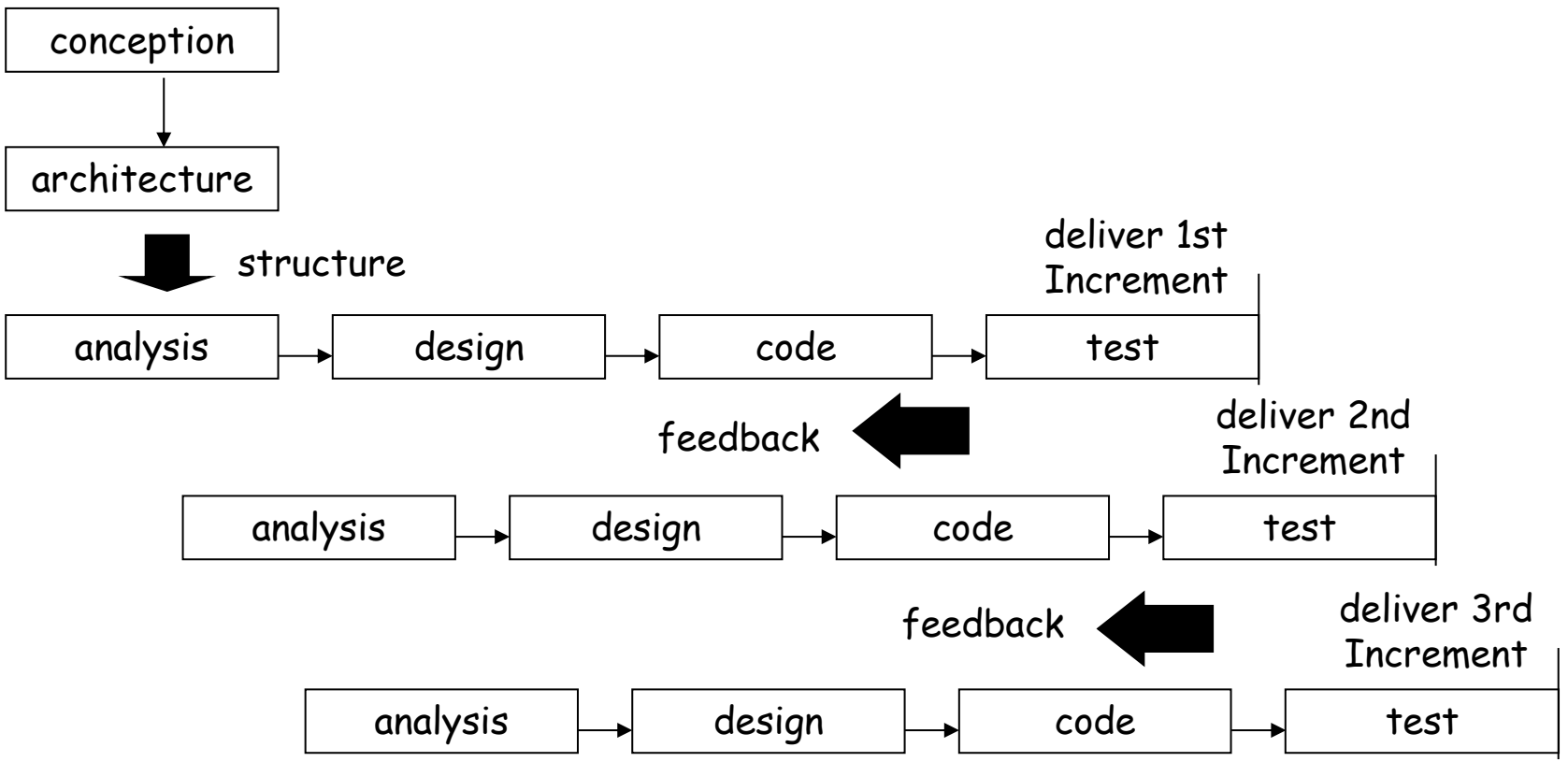
Incremental Development

The Incremental development model involves developing the system in an incremental fashion. The most important part of the system is first delivered and the other parts of the system are then delivered according to their importance.

Incremental development avoids the problems of constant change which characterize evolutionary prototyping. An overall system architecture is established early in the process to act as a framework.

Incremental development is more manageable than evolutionary prototyping as the normal software process standards are followed. Plans and documentation must be produced

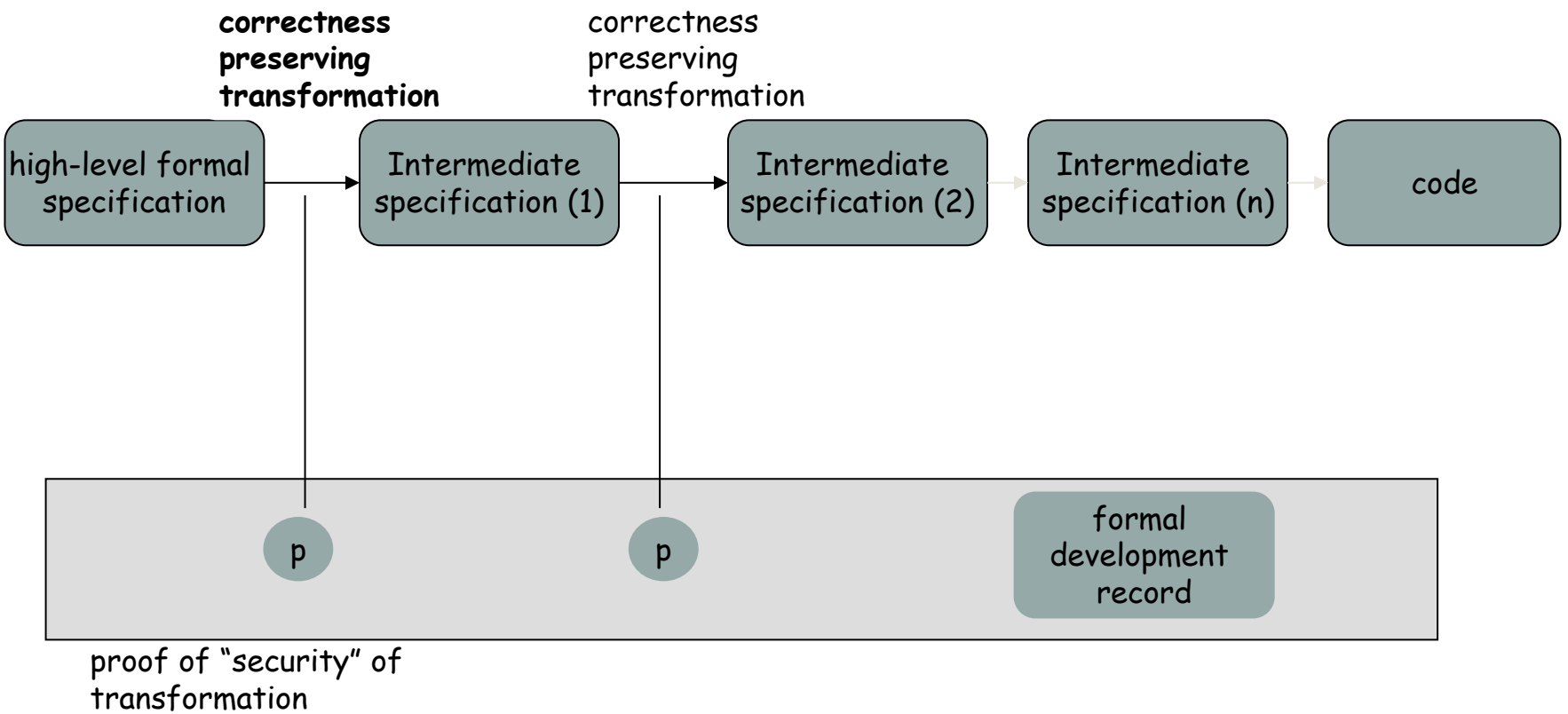
Incremental Model



Formal Specification

- Formal software specifications are mathematical entities and may be analyzed using mathematical methods. Specification consistency and completeness can be proved mathematically.
- Formal specifications may be automatically processed. Software tools can be used to build programs from formal specifications.
- The development of a formal specification provides insights into and an understanding of the software requirements and the software design.

Formal Development Model



Problems with formal development methods

- Many software engineers have not been trained in the techniques required to develop formal specifications.
- Customers may be unwilling to fund development activities that they cannot easily monitor.
- Software management is inherently conservative and is unwilling to adopt new techniques for which payoff is not obvious.
- Most of the effort in specification research has been concerned with the development of languages and their theoretical aspects rather than tools and methods.

The Spiral Model

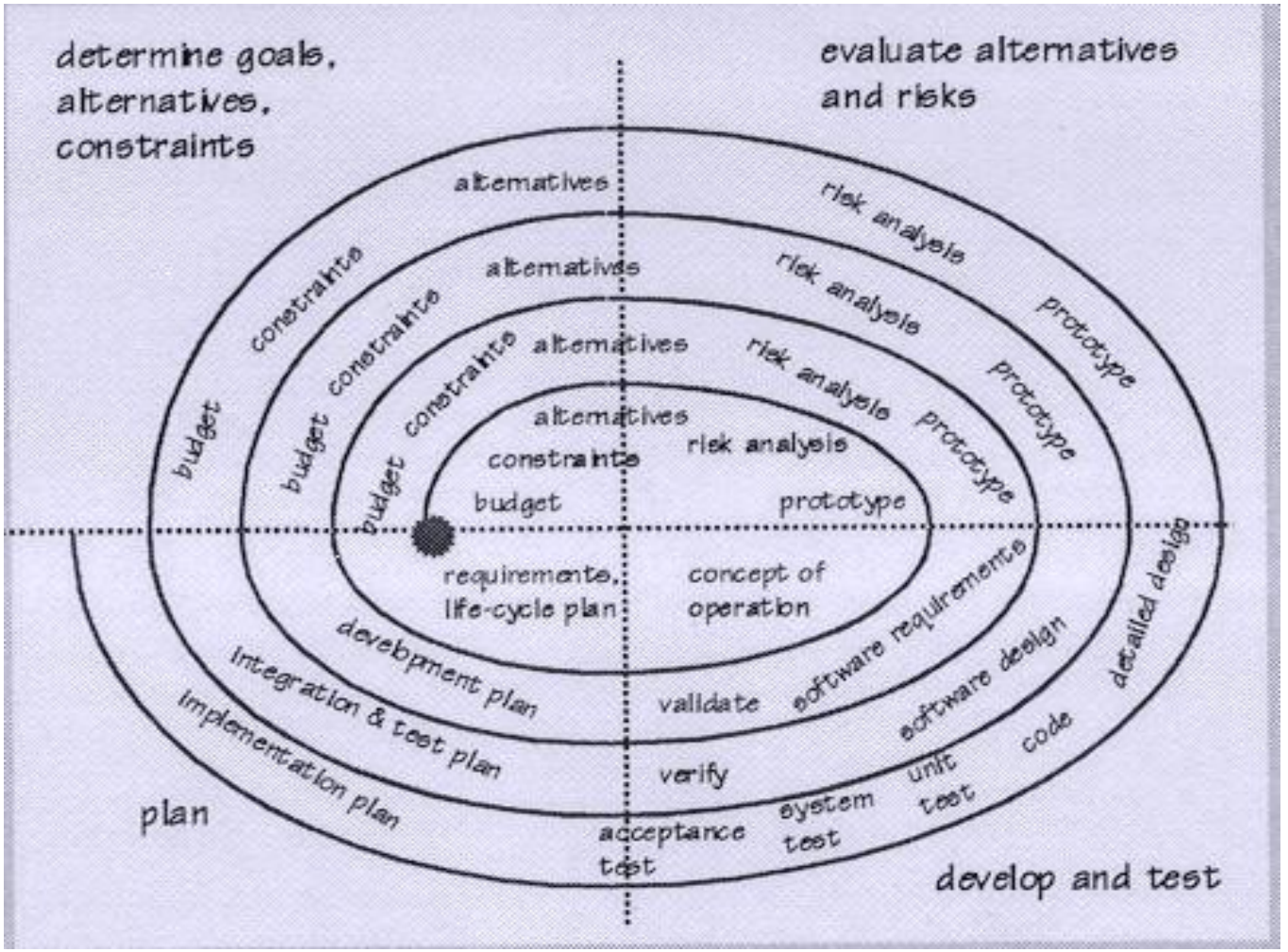
This model is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the linear sequential model.

Using the spiral model software is developed in a series of incremental releases. During early iterations, the incremental release might be a paper model or prototype.

The spiral model is divided into four main task regions

- Determine goals, alternatives , constraints
- Evaluate alternatives and risks
- Develop and test
- Plan

Spiral Model

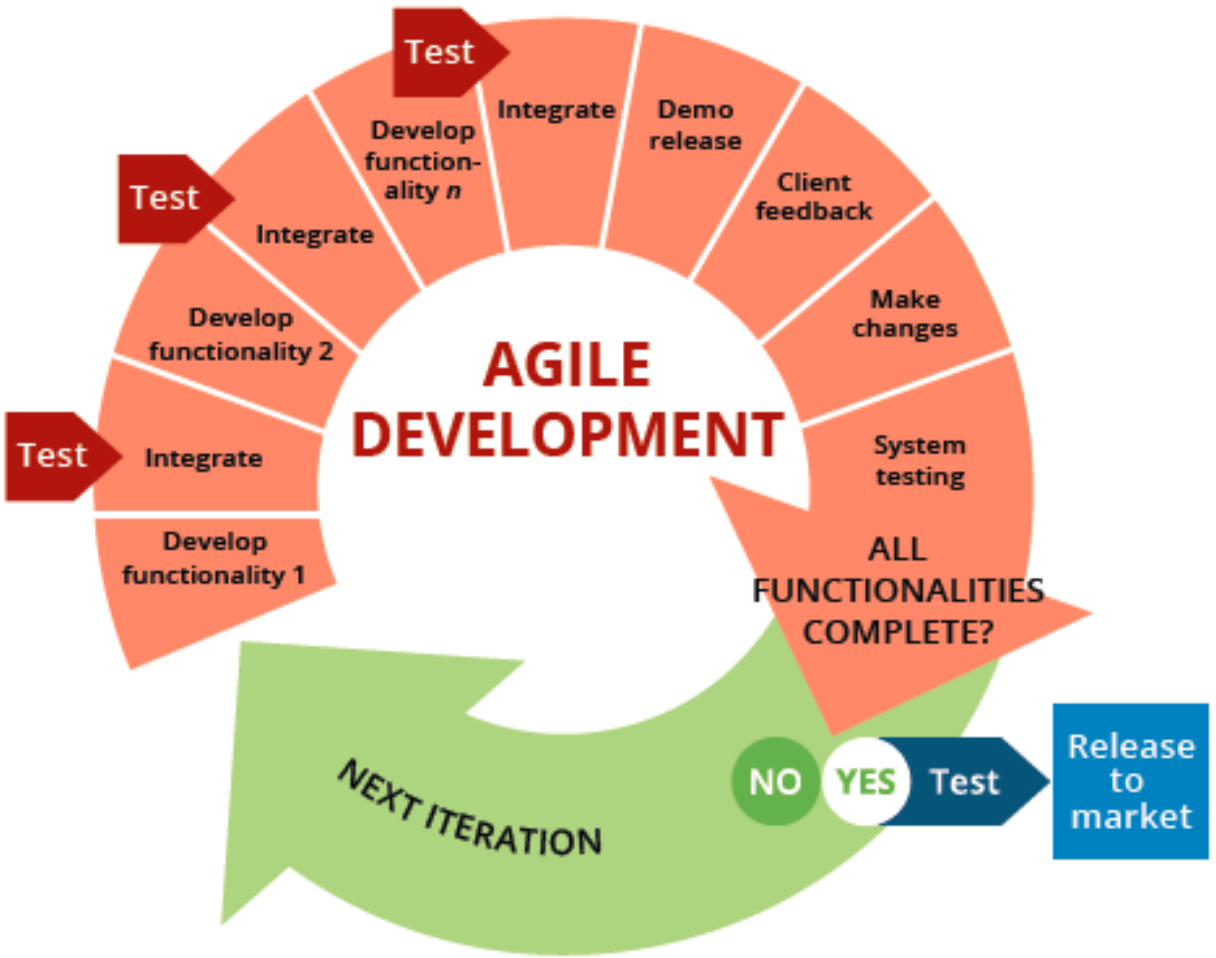


Agile Process

- Agile software engineering combines a philosophy and a set of development guidelines. The philosophy encourages the customer satisfaction and early incremental delivery of software;
- Iterative and Incremental Development: Breaks the project into smaller, manageable iterations (sprints), each delivering a functional product increment.
- Flexibility and Adaptability: Focuses on responding to changing requirements and priorities throughout the project.
- Customer-Centric Approach: Prioritizes customer feedback and ensures the final product aligns with user needs.

Agile Process

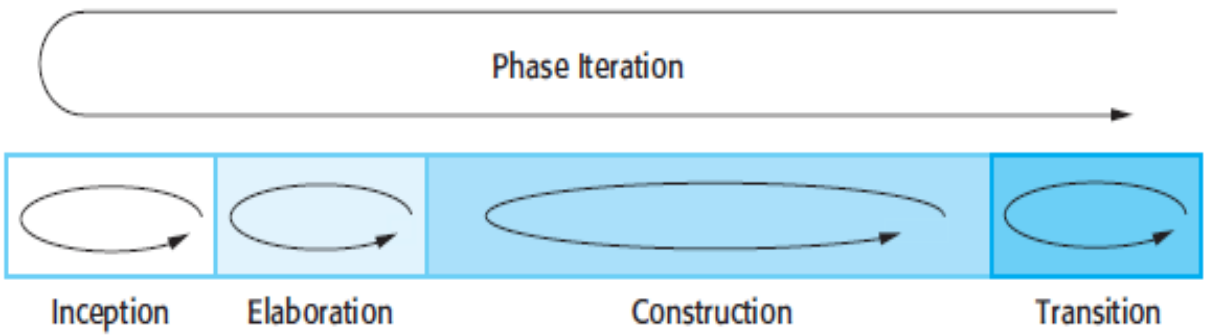
- An agile process adapt incrementally.
- To accomplish incremental adaptation, an agile team requires customer feedback.
- An effective tool to get customer feedback is an operational prototype or a portion of an operational system.
- Software increments must be delivered in short time periods so that the adaptation keep pace with the change.
- This iterative approach enables the customer to evaluate the software increment regularly and provide necessary feedback to the software team.


AGILE DEVELOPMENT
Develop functionality 1
Test
Integrate
Develop functionality 2
Test
Integrate
Develop function-ality n
Test
Integrate
Demo release
Client feedback
Make changes
System testing
ALL FUNCTIONALITIES COMPLETE?
NO
YES
Test
Release to market
NEXT ITERATION

The Rational Unified Process

- A modern generic process derived from the work on the UML and associated process.
- Brings together aspects of the 3 generic process models discussed previously.
- Normally described from 3 perspectives
  - A dynamic perspective that shows phases over time;
  - A static perspective that shows process activities;
  - A practice perspective that suggests good practice.

Phases in the Rational Unified Process



RUP phases

- **Inception**
  - Establish the business case for the system.
- **Elaboration**
  - Develop an understanding of the problem domain and the system architecture.
- **Construction**
  - System design, programming and testing.
- **Transition**
  - Deploy the system in its operating environment.

Static workflows in the RUP

Workflow	Description
Business modelling	The business processes are modelled using business use cases.
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements.
Analysis and design	A design model is created and documented using architectural models, component models, object models and sequence models.
Implementation	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.
Test	Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
Deployment	A product release is created, distributed to users and installed in their workplace.
Configuration and change management	This supporting workflow managed changes to the system (see Chapter 29).
Project management	This supporting workflow manages the system development (see Chapter 5).
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

Figure: Static workflows in the Rational Unified Process

RUP good practice

The practice perspective on the RUP describes good software engineering practices that are recommended for use in systems development

- Develop software iteratively

  Plan increments based on customer priorities and deliver highest priority increments first.
- Manage requirements

  Explicitly document customer requirements and keep track of changes to these requirements
- Use component-based architectures

  Organize the system architecture as a set of reusable components

RUP good practice *contd*..

- Visually model software

  Use graphical UML models to present static and dynamic views of the software

- Verify software quality

  Ensure that the software meets organizational quality standards

- Control changes to software

  Manage software changes using a change management system and configuration management tools.

Questions

Thank You

?