CS427a: Object-Oriented Programming

Design Patterns for Flexible and Reusable design

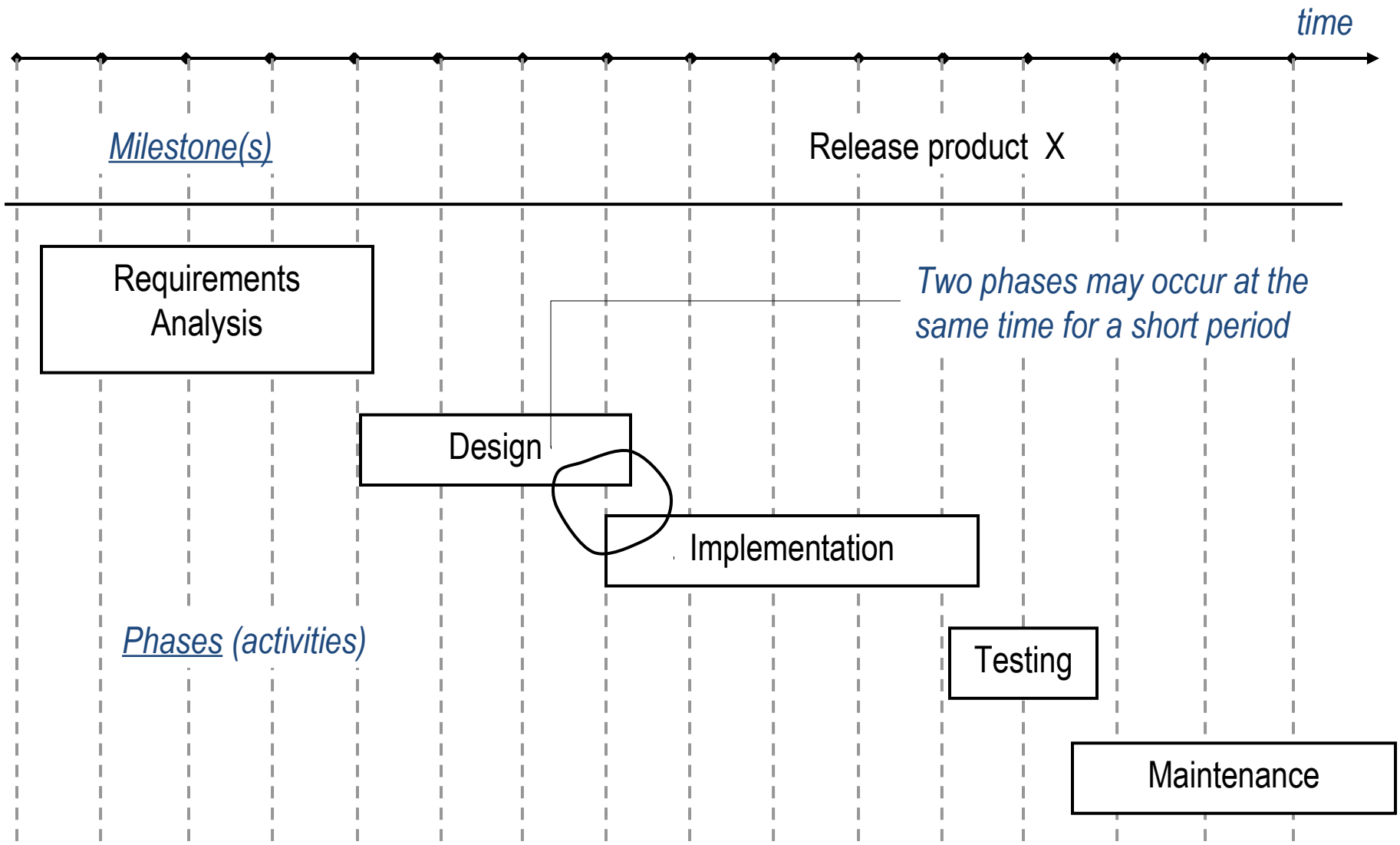
Michael J. Fischer
(from slides by Y. Richard Yang)

Lecture 22b
November 17, 2011

Reusability, Flexibility, and Maintainability

- One thing constant in software development is CHANGE
- For software that is used over a period of years, the cost of keeping it current in the face of changing needs often exceeds the cost of originally developing it.
- A key need in software design is the ability for maintenance and modification to keep abreast of changes.

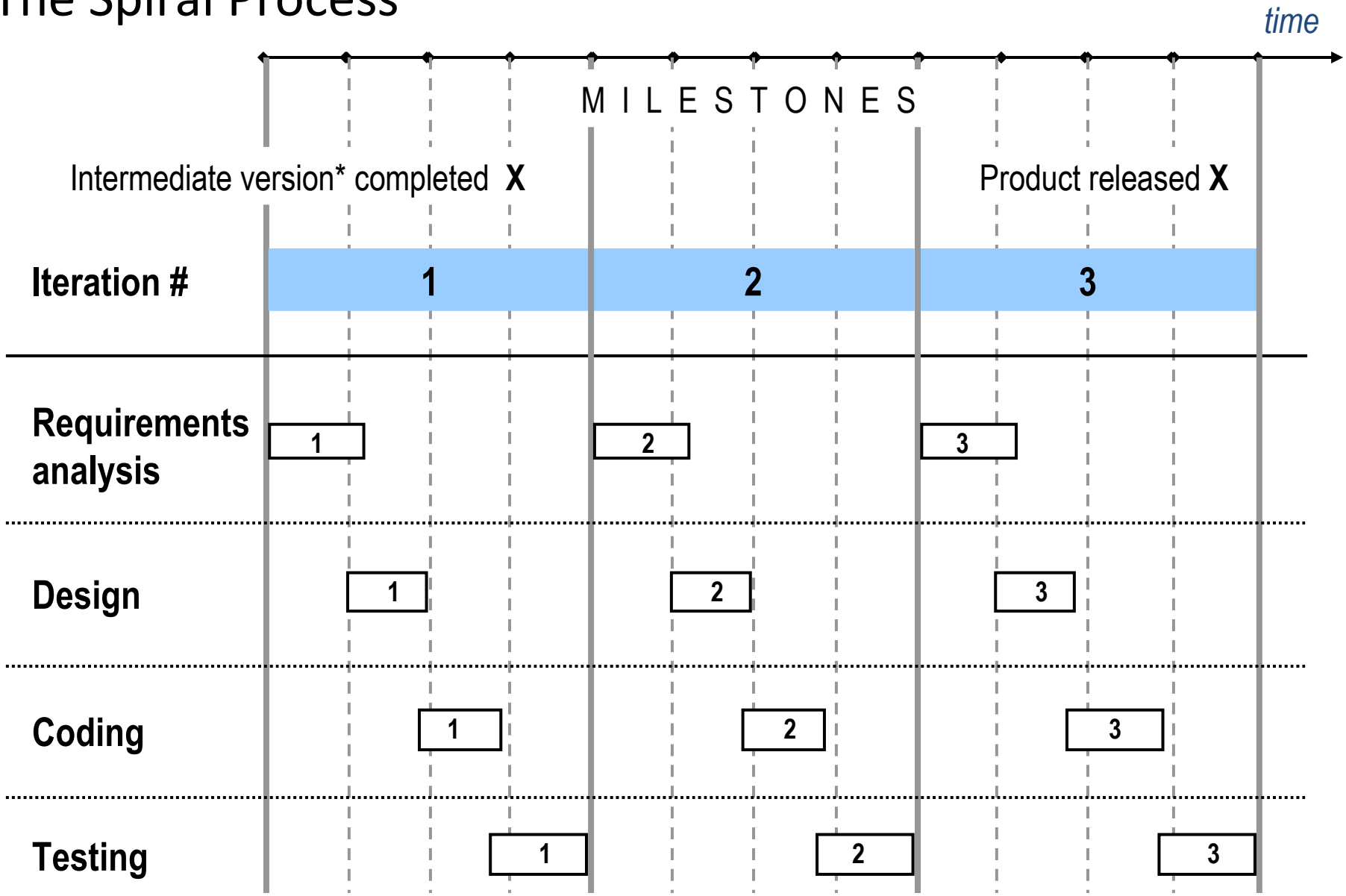
The Waterfall Software Process



Why a Pure Waterfall Process is Usually Not Practical

- *Don't know up front everything wanted and needed*
  - Usually hard to visualize every detail in advance
- To gain confidence in an estimate, we need to design and actually implement parts, especially the riskiest ones, this may probably lead to modify requirements as a result
- *We often need to execute intermediate builds*
  - Stakeholders need to gain confidence
  - Designers and developers need confirmation they're building what's needed and wanted
- *Team members can't be idle while the requirements are being completed*

The Spiral Process



*typically a prototype

Adapted from *Software Design: From Programming to Architecture* by Eric J. Braude (Wiley 2003), with permission.

Advantage of OO Design

OO systems exhibit recurring structures that promote

- Abstraction
- Modularity
- Flexibility
- Extensibility
- Elegance

Aspect of Reusability

- *Classes – in source code form*
  - Thus, we write *generic code* whenever possible
- *Assemblies of related classes*
  - A *toolkit* is a library of reusable classes designed to provide useful, general-purpose functionality.
    - E.g., C++ standard library, Boost
  - An *application framework* is a specific set of classes that cooperate closely with each other and together embody a reusable design for a category of problems.
    - E.g., Java APIs (`Applet`, `Thread`, etc), gtkmm
- Design pattern

Making a Class Re-usable

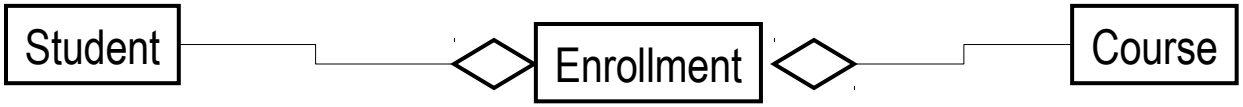
- Define a useful abstraction
  - attain broad applicability
- Reduce dependencies on other classes

…

Reducing Dependency Among Classes

Replace …

with …



Aspect of Flexibility

- Making small variation to existing functionality
- Adding new kinds of functionality
- Changing functionality

Some Techniques to Achieve Flexibility

Flexibility Aspect: ability to …	Some techniques
… create objects in variable configurations determined at runtime	"Creational" design patterns
… create variable trees of objects or other structures at runtime	"Structural" design patterns
… change, recombine, or otherwise capture the mutual behavior of a set of objects	"Behavioral" design patterns
… create and store a possibly complex object of a class.	Component
… configure objects of predefined complex classes – or sets of classes – so as to interact in many ways	Component

Roadmap

- We will focus on flexibility and reusability
  - It is important to remember that real systems also need to consider efficiency and robustness
- We will start with design patterns, and then look into the design of some OO libraries/toolkit/framework
- We will learn by examples:
  - *Example* is not another way to teach, it is the *only* way to teach. -- *Albert Einstein*

What is a Design Pattern

- Abstracts a recurring design structure
- Comprises class and/or object
  - dependencies
  - structures
  - interactions
  - conventions

- Distills design experience
- Names & specifies the design structure explicitly

- Language- & implementation-independent
  - A “micro-architecture”

UML/OMT Notation