


CSC444F: Software Engineering I

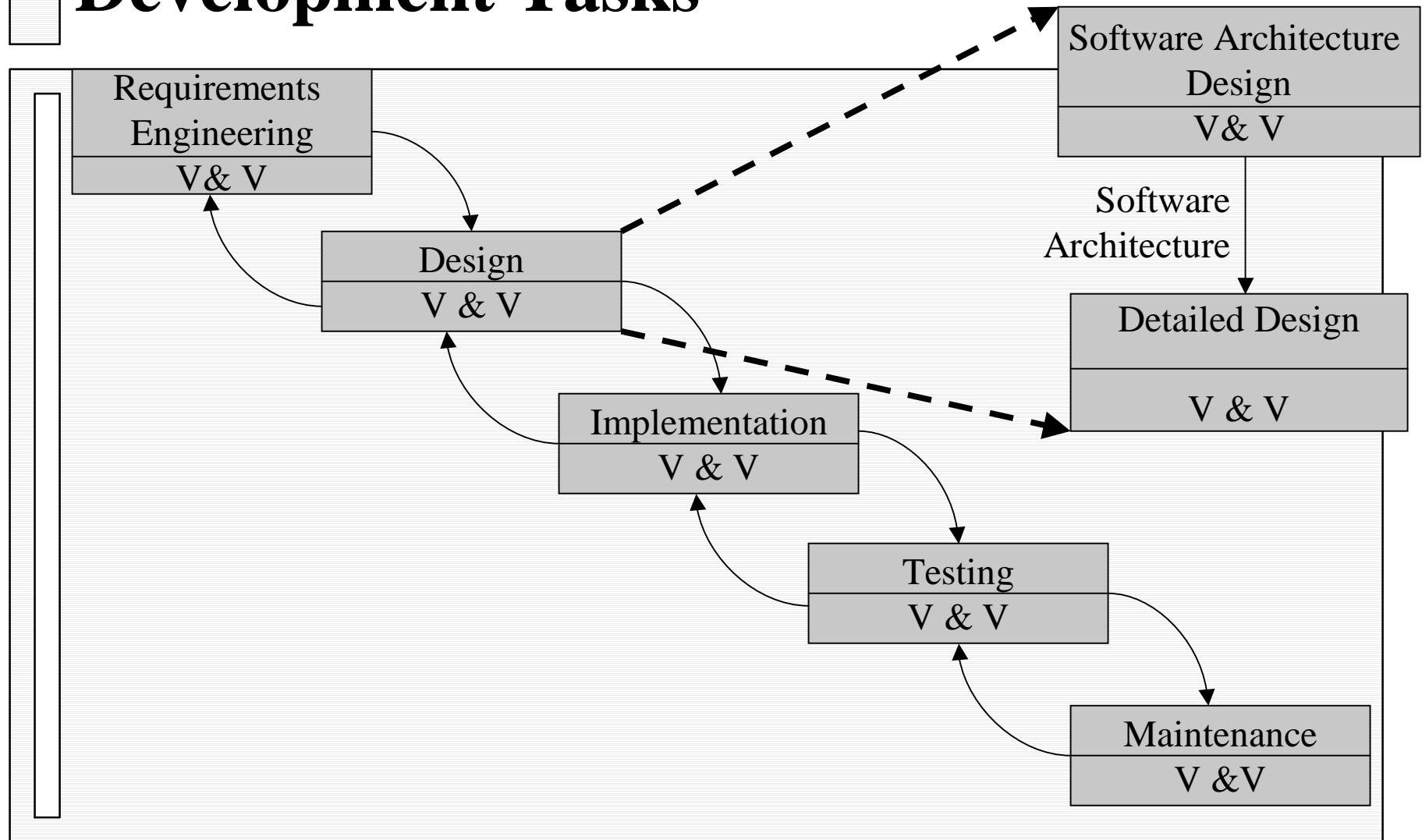
Mou Hu
mou.hu@utoronto.ca



Lecture 7: Software Architecture

- 
- What Is Software Architecture?
 - An Example: KWIC-Index Program
 - Software Architecture Styles
 - Reading: Chapter 10

Relation of Software Architecture to Development Tasks



What Is Software Architecture?

■ **The top-level decomposition of a system into major components together with a characterization of how these components interact, is called its software architecture.**

■ **Two main aspects of software architecture:**

- It provides a **design plan** - a blueprint - of a system, which is a vehicle for communication among stakeholders and captures early design decisions.
- It is an **abstraction** to help manage the complexity. It is also a basis for reuse.

■ **Four views of software architecture**

- **Conceptual view** (logical view) - design elements
- **Implementation view** (module view) - modules, packages, and layers
- **Process view** (execution view) - tasks, processes, runtime elements
- **Deployment view** (code view) - source components

Software Architecture Terminology

- **Architecture Style** or **Architecture Pattern** (usually not domain specific)
e.g. pipes-and-filters
- **Reference Architecture** or **Domain-specific Software Architecture** (applies to a particular domain)
e.g. process control architecture
- **Product-line Architecture** (applies to a set of products within an organization)
- **Software Architecture** (applies to a system or product)
- **Design Pattern** (relevant to architecture or detailed design)
- **Framework** (a hybrid of architecture-level information and implementation)

An Example: KWIC-Index Program

- KWIC = Key Word In Context

Task is to build a contextualized index for the text

Input is a set of lines of text

Output is the set of all circular shifts of all lines, in alphabetical order

- Four possible different architectures:

1) shared data model

2) data abstraction model

3) event based model

4) pipes-and-filters model

- Evaluation of the architectures with respect to aspects:

Change in data representation; Change in functionality; Reusability;

The degree to which modules can be implemented independently

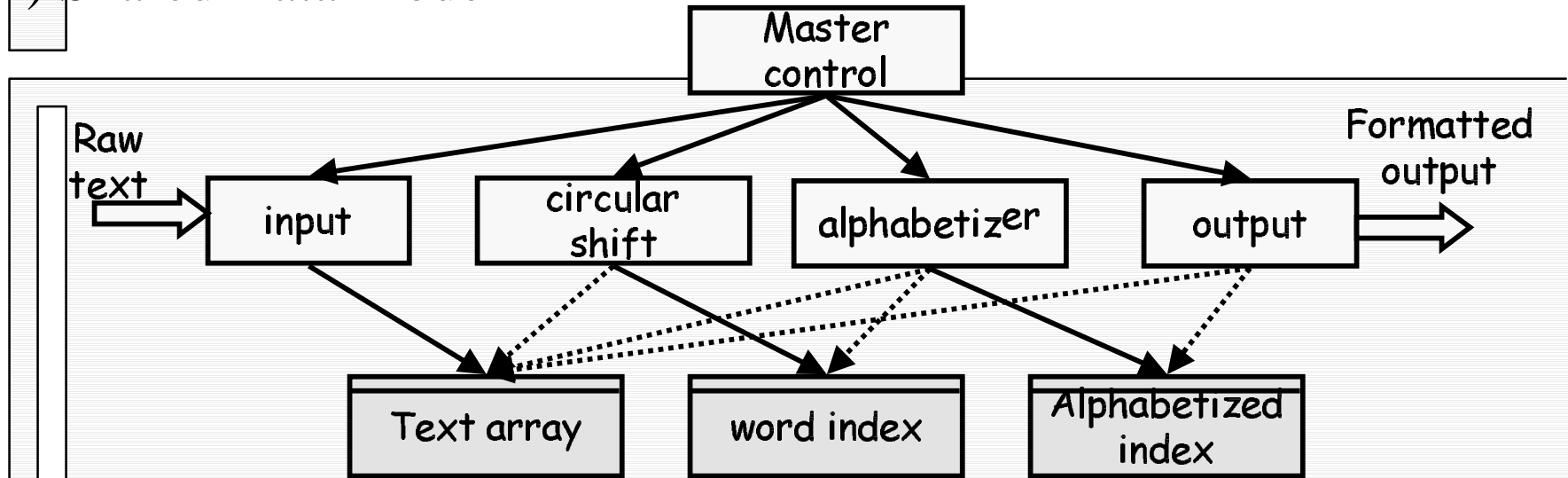
- Evaluation conclusion:

Architecture 1) good for adding functionality; poor for change in data representation, reusability, and independent development

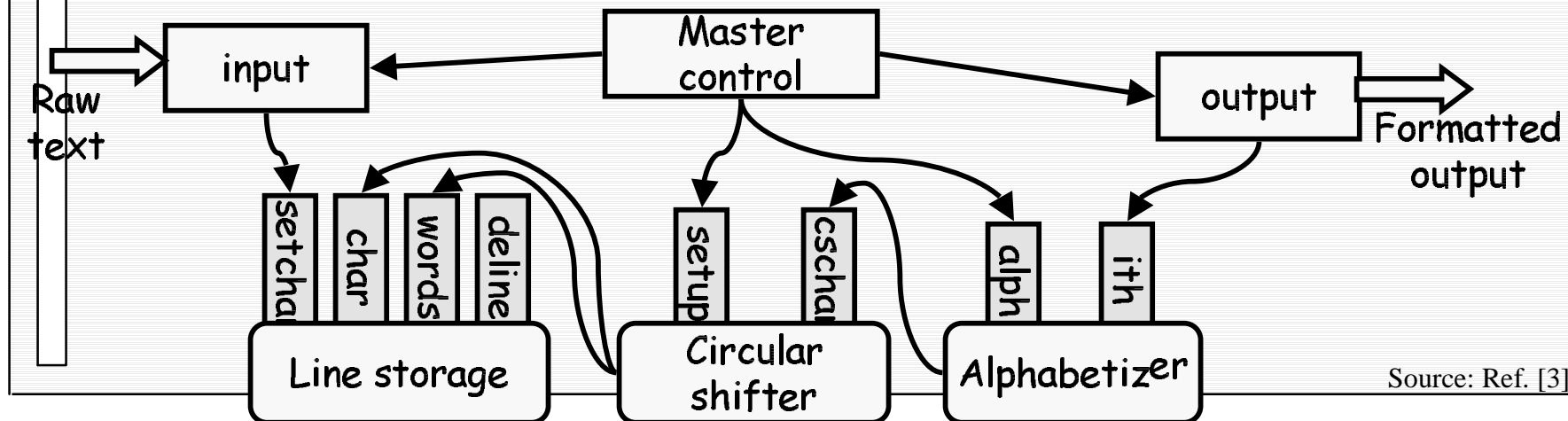
2) good for changing data representation, reusability, and independent development; poor for change in functionality

KWIC Architecture Solutions

1) Shared Data Model



2) Data Abstraction Model

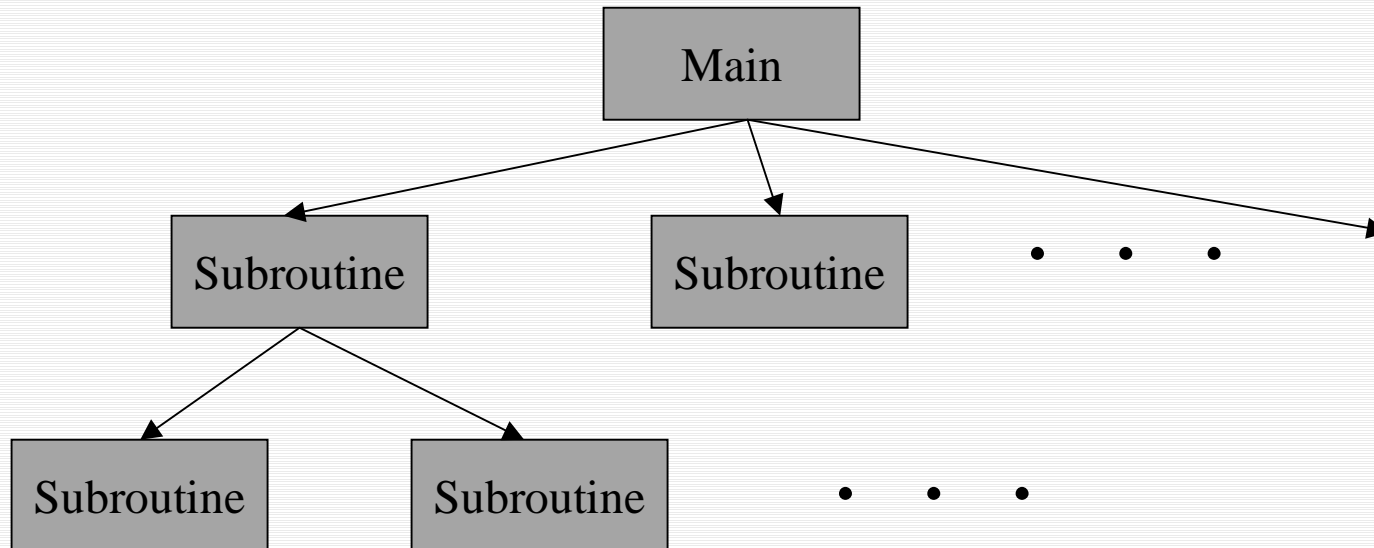


Source: Ref. [3]

Components and Connectors

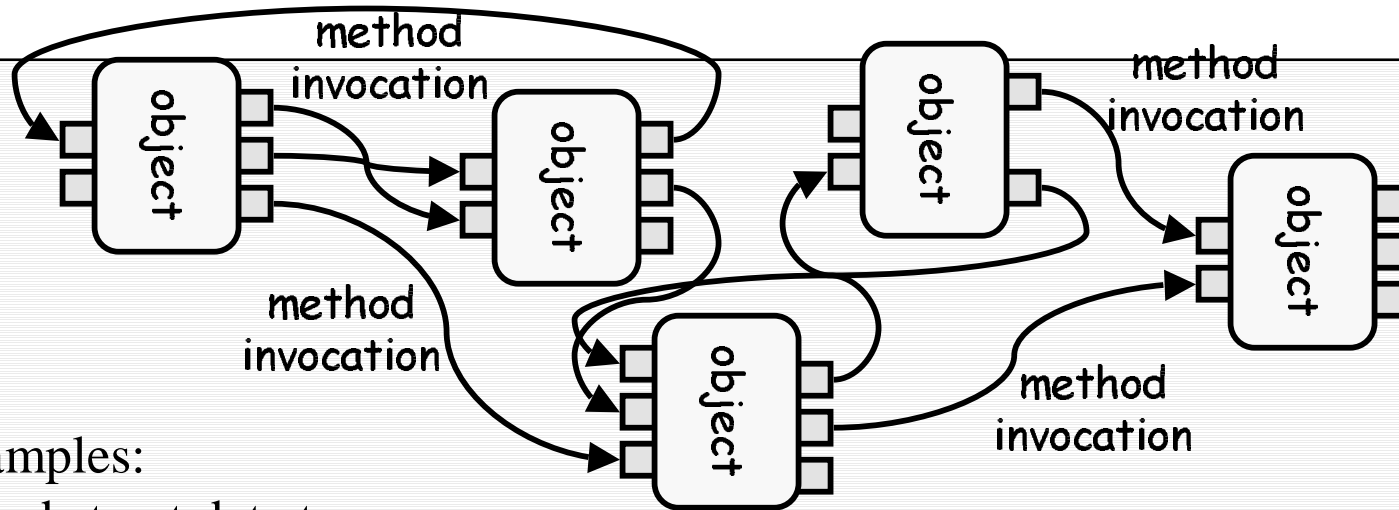
- The major elements of a software architecture are components and connectors.
- Components are the building blocks of a software architecture.
- Some component types: computational, memory, manager, controller.
- The connectors describe how components interact.
- Some connector types: procedure call, data flow, implicit invocation, message passing, shared data, instantiation.

Architecture Style 1: Main Program with Subroutines



- This style is a natural outcome of a functional decomposition of a system.
- Components: procedures, which may have their own local data, and global data, which may be viewed as residing in the main.
- Connectors: Procedure call and shared access to global data.

Architecture Style 3: Abstract Data Types



Examples:

- abstract data types
- object broker systems (e.g. CORBA)

Interesting properties

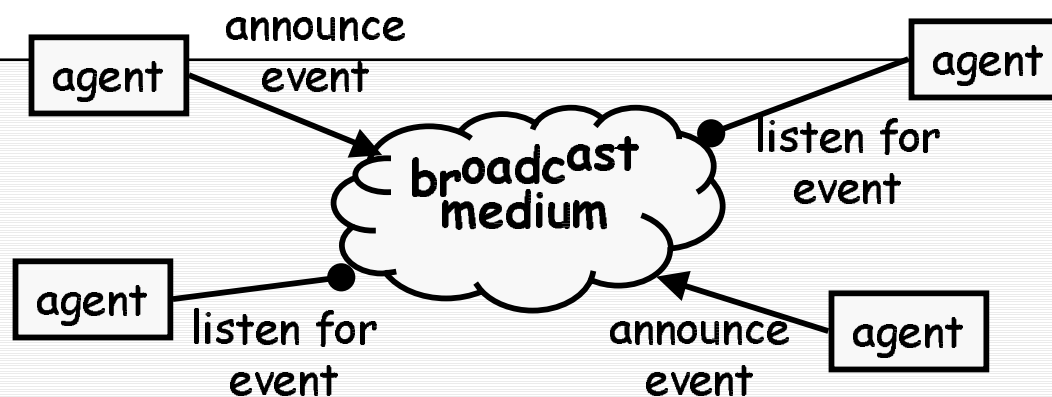
- data hiding (internal data representations are not visible to clients)
- can decompose problems into sets of interacting agents

Disadvantages

- objects must know the identity of objects they wish to interact with

Source: Ref. [3]

Architecture Style 4: Event Based (Implicit Invocation)



Examples

- debugging systems (listen for particular breakpoints)
- database management systems (for data integrity checking)
- graphical user interfaces

Interesting properties

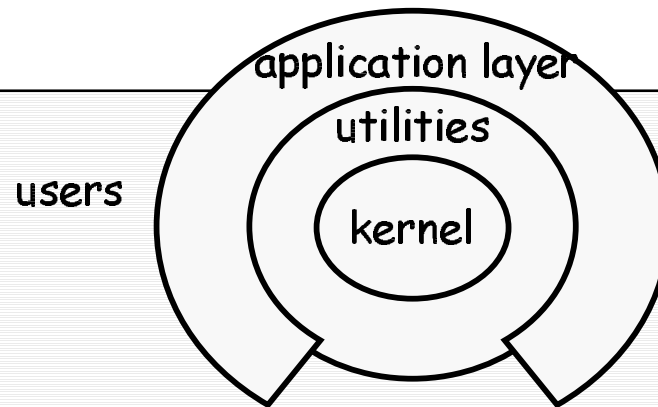
- announcers of events don't need to know who will handle the event
- Supports re-use, and evolution of systems (add new agents easily)

Disadvantages

- Components have no control over ordering of computations

Source: Ref. [3]

Architecture Style 5: Layered Systems



Examples

Operating Systems

Communication protocols

Interesting properties

Support increasing levels of abstraction during design

Support enhancement (add functionality) and re-use

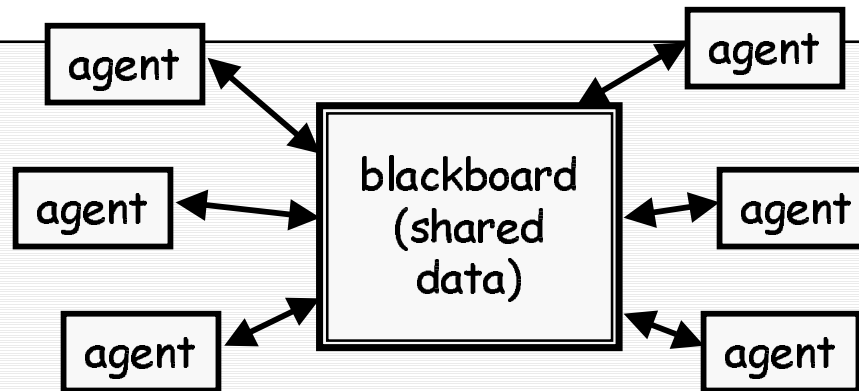
Can define standard layer interfaces

Disadvantages

May not be able to identify (clean) layers

Source: Ref. [3]

Architecture Style 6: Repositories



Examples

databases

blackboard expert systems

programming environments

Interesting properties

can choose where the locus of control is (agents, blackboard, both)

reduce the need to duplicate complex data

Disadvantages

blackboard becomes a bottleneck

Source: Ref. [3]

References

- [1] Hans van Vliet, “Software Engineering: Principles and Practice”, John Wiley and Sons, Ltd., 2000. Chapter 10.
- [2] Christine Hofmeister, Robert Nord, and Dilip Soni, “Applied Software Architecture”, Addison Wesley Longman, Inc., 2000.
- [3] Steve Easterbrook, “Lecture Notes”, University of Toronto, 2001.