

CSC444F: Software Engineering I

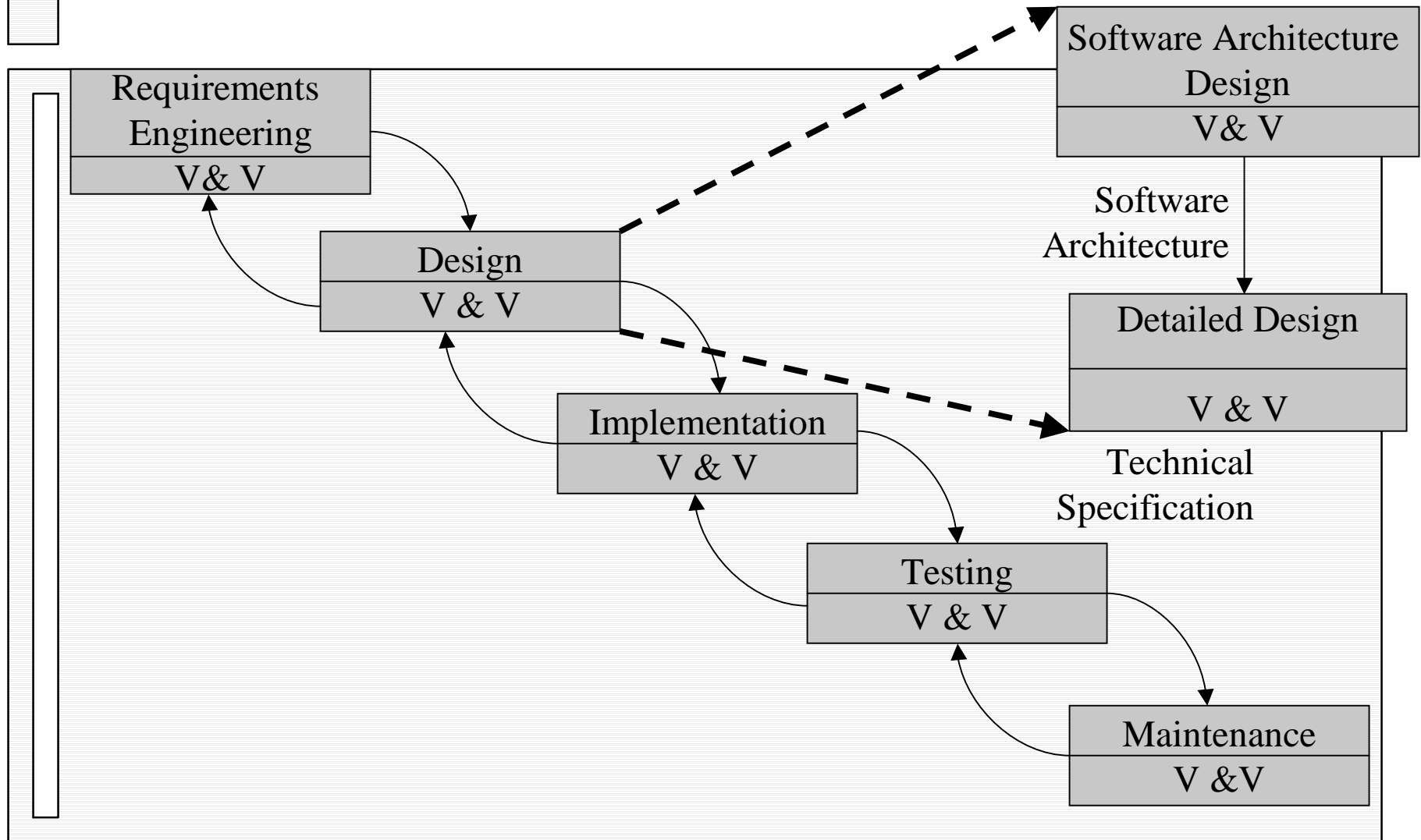
Mou Hu
mou.hu@utoronto.ca

Lecture 8:

Software Design (1)

- Design Considerations:
 - Decomposition
 - Abstraction
 - Complexity
- Design Method 1 - Functional Decomposition
- Design Documentation
- Reading: Chapter 11

Design Phase (Non-OO)



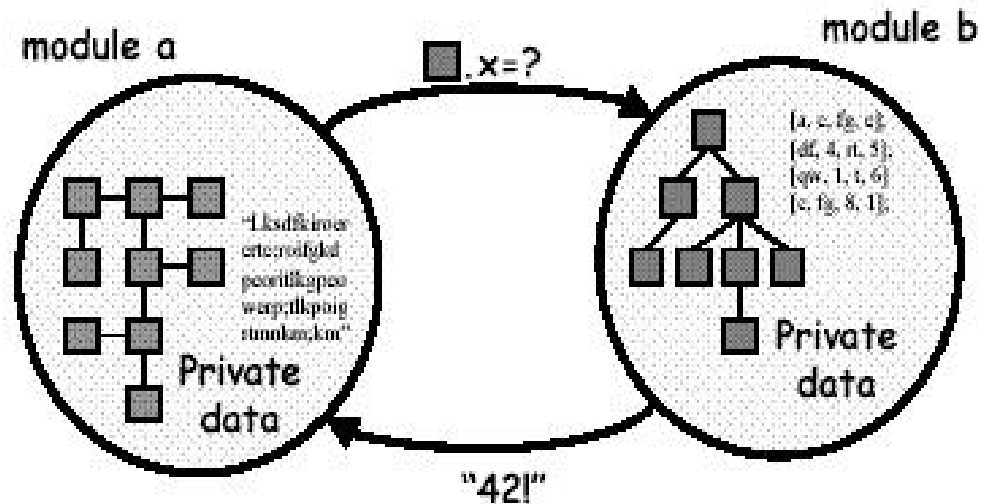
Decomposition

- **Large systems can be developed with “divide and conquer”**
- **Design can be formulated as decomposing the system into parts so that:**
 - **Each part is at (roughly) the same level of detail and has lower complexity than the whole system**
 - **Each part can be developed independently**
 - **These parts can be combined to solve the user’s problem**
- **Advantages:**
 - **Different people can work on different parts**
 - **Parallelization may be possible**
 - **Maintenance is easier**

Good Decomposition - Modularity

■ A good decomposition minimizes dependencies between components

- **Loose coupling** - a measure of inter-component connectivity
- **Strong cohesion** - a measure of how well the contents of a component go together
- **Information hiding** - having modules keep their data private; provide limited access procedures; this reduces coupling



Source: Ref. [3]

Abstraction

- **Abstraction is the main tool used in reasoning about software**
- **It allows you to:**
 - **ignore inconvenient detail**
 - **treat different entities as though they are the same**
 - **simplify many types of analysis**
- **Two types of abstraction:**
 - **Procedural abstraction**
 - **Data abstraction**

Procedural Abstractions

- A procedure maps from input to output parameters

it may modify its parameters; it may have side effects;
it may return a result

- A procedural abstraction (“specification”):
describes what a procedure does, ignores how it does it;
different implementations of the abstraction can differ over
details; one implementation can be substituted for another

- Example:

```
procedure search (a: list of int, x:int) returns i:int  
requires: a is sorted in ascending order  
effects: If x is in a, i is the index of an occurrence  
         of x in a, so that a[i]=x otherwise i is -1
```

Source: Ref. [3]

Source: Ref. [3]

Data Abstraction

■ Defining Data Abstractions: The abstraction should

name the data type;

list its operations;

describe the data

abstraction in English;

say whether it's

mutable or not;

give a procedural

abstraction for each

operation;

the abstraction only

lists the “public”

operations,

there may be other

“private” procedures

hidden inside...

```
/*
datatype set has operators create, insert, delete,
member, size, union, intersection.
overview:
sets are unbounded mathematical sets of integers.
They are mutable: insert and delete are the
mutation operations.
operations:
procedure create () returns set
effects: x is a new empty set

procedure insert (set s, int x) returns null
effects: adds x to the set s such that  $s' = s \cup \{x\}$ 

procedure delete (set s, int x) returns null
requires:  $x \in s$ 
effects:  $s' = s - \{x\}$ 

... (etc) ... */
```

Source: Ref. [3]

Complexity

- Complexity refers to attributes of the software that affect the effort needed to construct or change a piece of software.

- Size-based complexity metrics.

- LOC - lines of code

- “Software science” (Halstead’s method) - uses the number of operators and operands in a piece of software to derive entities such as level of abstraction, programming effort, and estimated programming time.

- Structure-based complexity metrics.

- McCabe’s cyclomatic complexity:

$$CV = e - n + p + 1$$

(e - edges, n - nodes, p - connected components in a control graph)

Design Method 1: Functional Decomposition

- This is a general approach.
- Top-down design: starting from the main user functions, we work down decomposing functions into sub-functions.
- Bottom-up design: starting from a set of available base functions, we proceed towards the requirements specification.
- In practice, we can opt for a particular architectural style and let the style guide the decomposition.

Design Documentation

- IEEE Standard 1016 describes an information model. The entities in this model are the components identified during the design stage.
- The attributes of a component: Identification, Type, Purpose, Function, Subordinates, Dependencies, Interfaces, Resources, Processing, Data.
- Different views on the design serve different groups of users.
 - Decomposition - decomposition of the system into modules
 - Dependencies - relations between modules and between resources
 - Interface - how to use modules
 - Detail - internal details of modules
- Different notations can be used to describe the design.

References

- [1] Hans van Vliet, “Software Engineering: Principles and Practice”, John Wiley and Sons, Ltd., 2000. Chapter 11.
- [2] Stephen R. Schach, “Object-Oriented and Classical Software Engineering”, McGraw-Hill Companies, Inc., 2002.
- [3] Steve Easterbrook, “Lecture Notes”, University of Toronto, 2001.