

COMP2110/2510/6444 Software Design 2007

lecture 8 **Software Architecture 1 of 2**

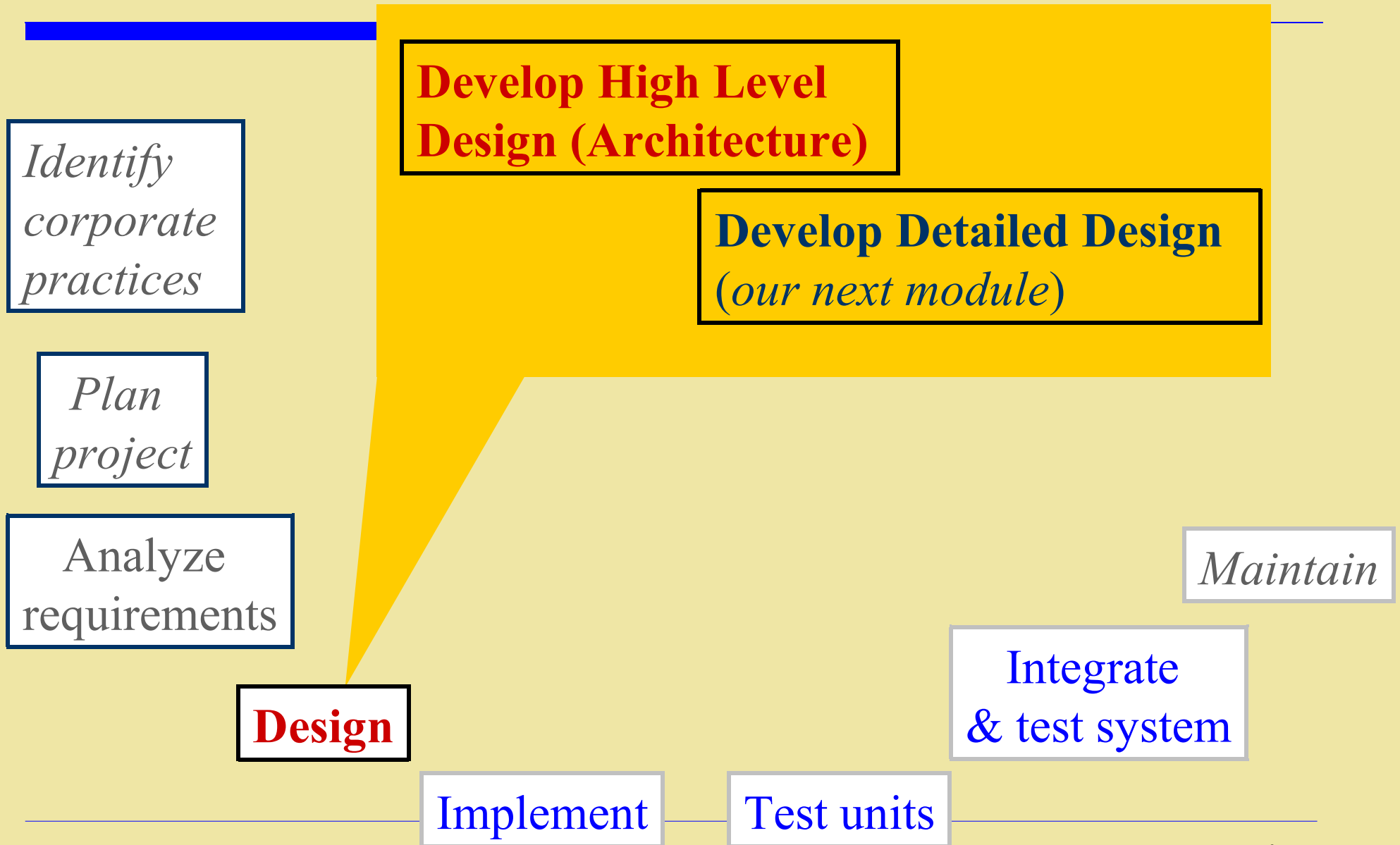
(*Design* lecture 3 of 6)

Goal of this 2 lecture module:

a descriptive understanding of “architecture”
(*not* the detailed application of a big library of examples).

2. What is architecture (high level) design vs detailed design
3. some existing software architectures

Software Engineering Roadmap (*Braude SE chap 5*)



One way to ... Begin Selecting a Basic Architecture

1. Develop a mental model of the application at a high level.

- as if it were a small application

e.g., personal finance application ...

... “works by receiving money or paying out money, in any order, controlled through a user interface”.

Note: To use established architectures, see the rest of this chapter

2. Decompose into the required components.

- look for high cohesion & low coupling

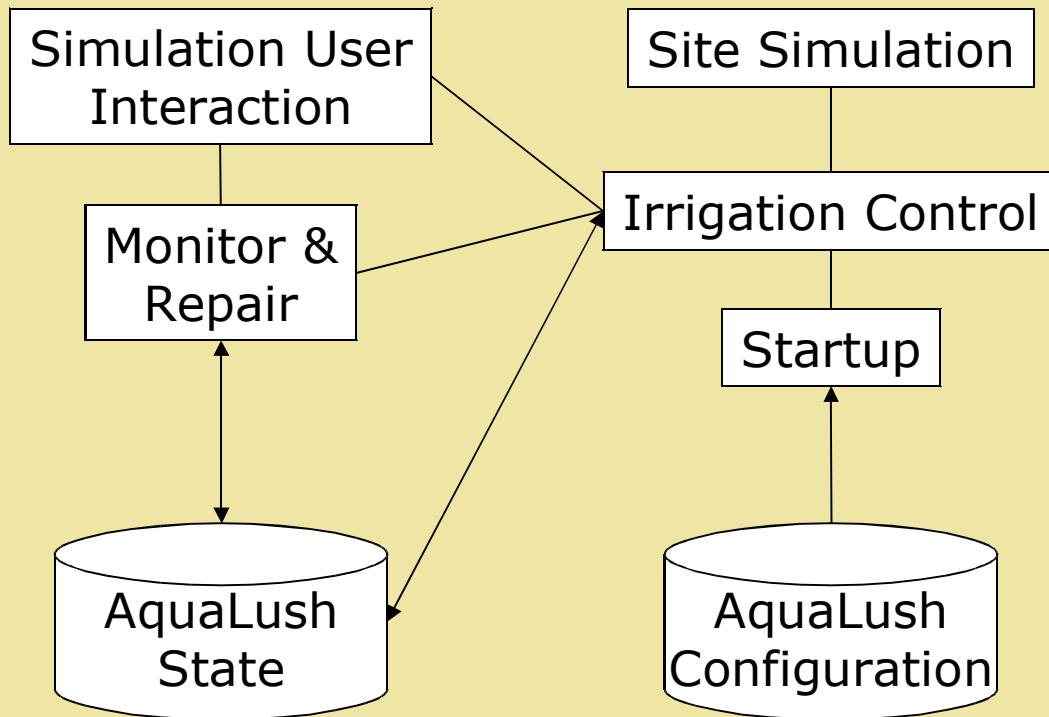
e.g., personal finance application ...

... decomposes into Assets, Suppliers, & Interface.

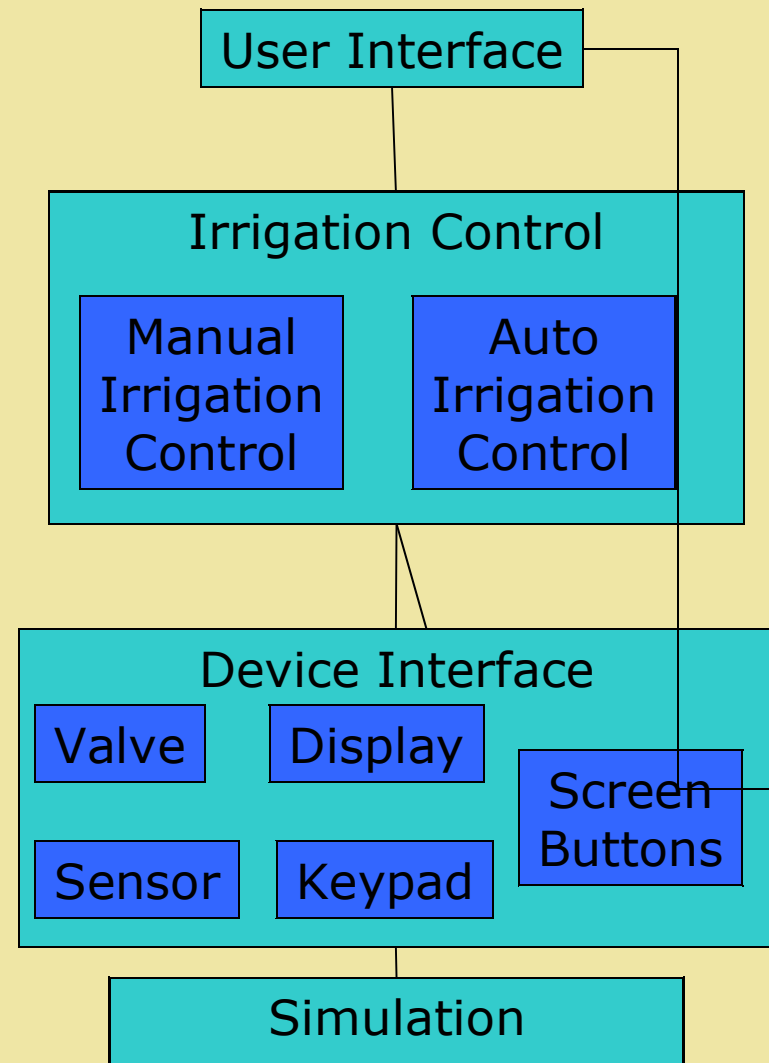
3. Repeat this process for the components.

Software Architecture design

Fox Fig 10-1-11 AquaLush Simulation Modular Decomposition



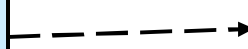
Fox Fig 10-1-6 AquaLush Simulation Functional Decomposition



Use-case model: “do this”

Business use case

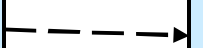
Use case



elaborated by ...

Sequence diagram

Scenarios



Class model: “with ...”

package

consists of ...

class

methods



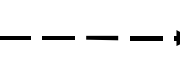
Target Application

for Design

Component model: “how”

Component

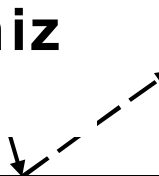
Data flow



organize by ...

Local data flow

Sub-component



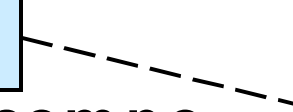
State model: “when”

States

decompose into ...

Substates

Transitions



Models

Software Architecture – in general

The decomposition of the design into groups of abstract modules (clusters of classes)

- modules are concepts, with names
 - account management, playing field, characters
- Quality
 - (1) coherence and coupling
 - stronger/more relationships between classes inside the module boundary than going in and out through the boundary
 - (2) potential for generality and reuse in other projects
 - E.g. PlayingField in Tetris

Elements of software architecture

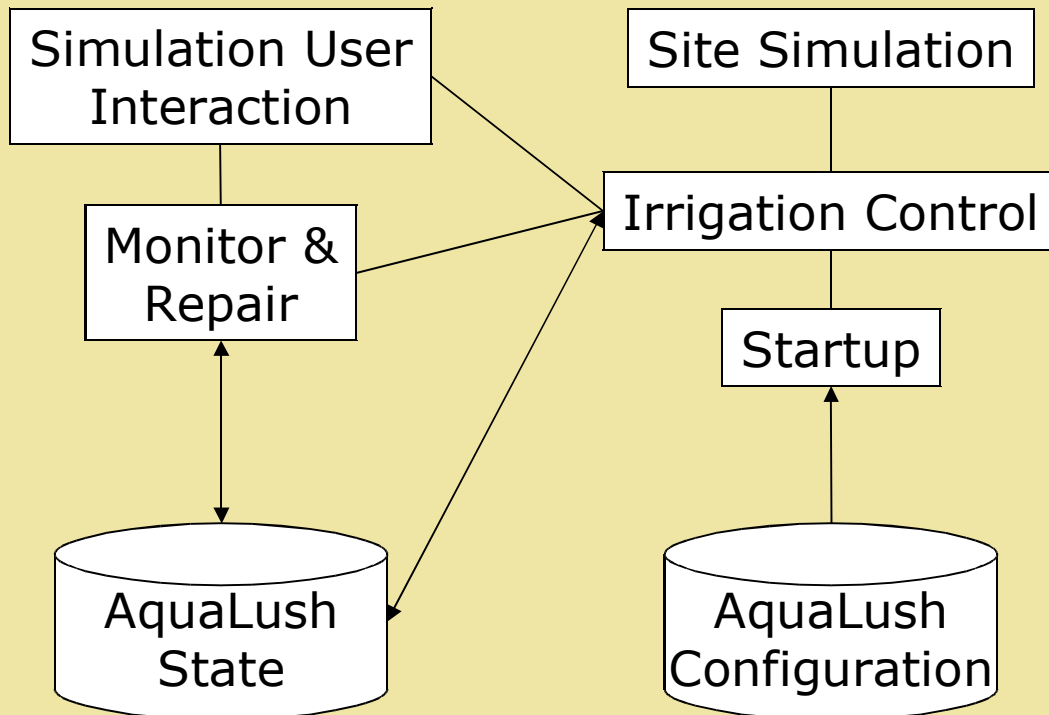
- description of elements from which software systems are built
- interactions between those elements
- patterns that guide their composition
- constraints on those patterns

In general, a system is defined in terms of

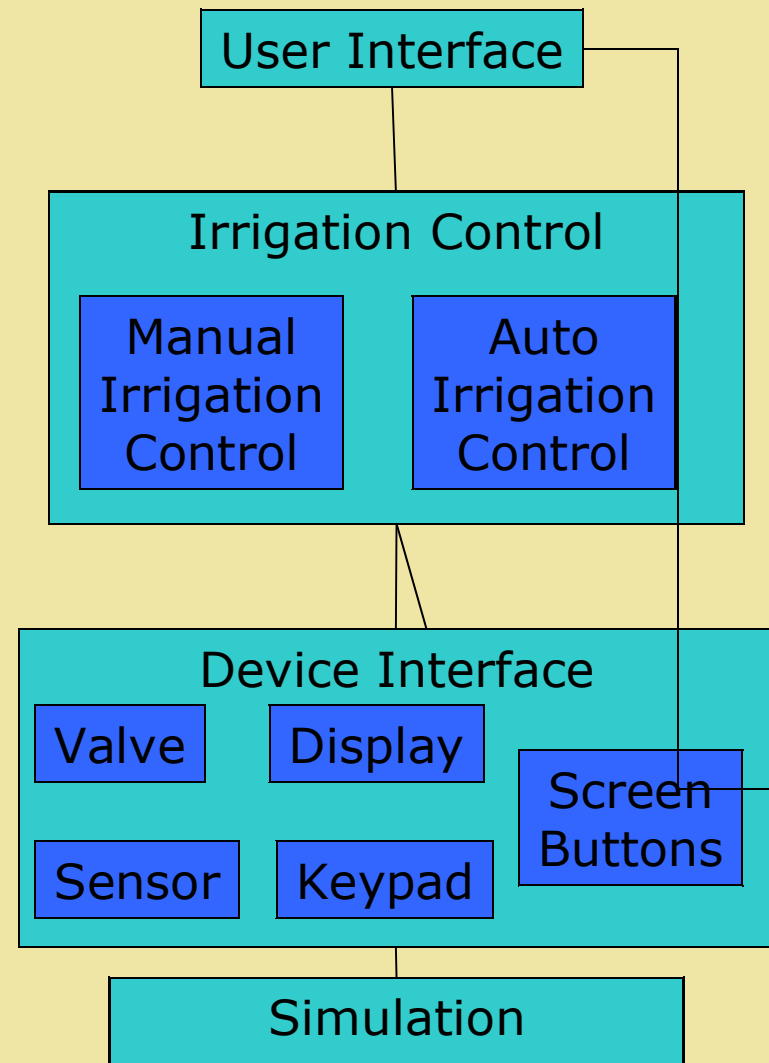
- a collection of components
- interactions between those components

Software Architecture design (recap)

Fox Fig 10-1-11 AquaLush Simulation Modular Decomposition



Fox Fig 10-1-6 AquaLush Simulation Functional Decomposition



Existing software architectures

- There are several well known architectures that are widely used, named, discussed
- Leading example: Model-View-Controller for a spreadsheet
 - model: the cells and values and formulae
 - views:
 1. a rectangular window subset of cell values
 2. a chart of any rectangular subset of cells
 - controller: the evaluation function

Existing software architectures (2)

Important examples

- dataflow systems
- call-and-return systems
- independent components
- virtual machines
- data-centred systems

from Mary Shaw and David Garlan, Software Architecture: Perspectives on an Emerging Discipline, Prentice-Hall 1996 fig 2.1

- *See chapter 15 Fox for a set of “architectural styles”*

Existing software architectures (3)

- dataflow systems
 - batch sequential
 - pipes and filters
- call-and-return systems
 - main program and subroutine
 - OO systems
 - hierarchical layers

*example coming up
in next lecture*

Existing software architectures (4)

- independent components
 - communicating systems
 - event systems
- virtual machines
 - interpreters
 - rule-based systems
- data-centred systems
 - databases
 - hypertext systems
 - blackboard systems

Why is there a choice of architectures?

- different architectures can be more or less:
 - efficient
 - robust to changes,
 - robust to hardware and network failures
 - costly to implement
- e.g. *robust to changes* in
 - algorithms
 - data representations
 - enhancements to the specification and functionality
 - performance needs
 - reusability
- design is always a choice between many factors which the designer must weigh up – there is never only one “correct” solution

Fox's view of Software Architecture design

- 9.1 Intro to Architectural Design
- 9.2 Notation: Boxes and Arrows
- [9.3, 9.4 *UML notation: package, component, deployment*] not in this course
- 10.1 Generating Software Architectures
- 10.2 Evaluating and Selecting S/W Archs.

Braude's Software Design: view of Software Architecture

- *ch 14.1 (meaning of architecture)*
- *ch 14.2 (models)*
- *ch 14.3 architecture: goals, modularization, cohesion and coupling, (patterns – more later), standard architectures*
- *ch 14.4 frameworks – not for comp2110*