

# INF 117 Project in Software Engineering

Lecture Notes - Winter Quarter, 2008

Michele Rousseau

Set 6 - System Architecture

\* Some slides adapted from Ping Wang

## Announcements

⌘ Due 2/15

- Design Iteration #3 (Final)
- Customer Milestone (Design Approved)

⌘ Due 2/18

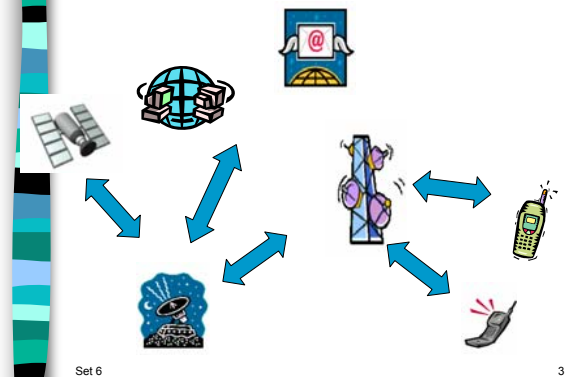
- Code Iteration #1
- Project Plan #3

⌘ Can submit Subjective Assessments online (but only if prior to the due date)

Set 6

2

## System Architecture



Set 6

3

## Software Architecture: Essentials

⌘ Components

- What are the main parts?
- What aspects of the requirements do they correspond to? Where did they come from?
- Examples: filters, databases, GUIs, interpreters

⌘ Connections

- How do components communicate?
- Examples: procedure calls, messages, pipes, event broadcast

⌘ Constraints (including constraints on change)

How is it all organized?

Set 6

4

## Architectural design process

⌘ System structuring

- Decompose the system into principal sub-systems
- identify communications between them

⌘ Control modelling

- A model of the control relationships between the different parts of the system

⌘ Modular decomposition

- The identified sub-systems are decomposed into modules

Set 6

5

## Architecture Design: Advanced

⌘ Architectural styles

- Restrict the way in which components can be connected
- Prescribe patterns of interaction
- Promote fundamental principles
- Common styles: layered, client server, etc

⌘ Architecture description

- Boxes and arrows
- UML
- Architecture description languages

Set 6

6

## From Architecture to Modules

- Repeat the design process
  - Design the internal architecture of a component
  - Define the purpose of each module
  - Define the provided interface of each module
  - Define the required interface of each module
- Do this over and over again
  - Until each module has...
    - a simple, well-defined internal architecture
    - a simple, well-defined purpose
    - a simple, well-defined provided interface
    - a simple, well-defined required interface
- Until all modules "hook up"

Set 6 7

## Some Principles

- Rigor
  - ensures all requirements are addressed
- Separation of concerns
  - Modularity
    - allows work in isolation because components are independent of each other
    - decompose a complex system into less complex sub-systems; divide and conquer
    - (re-)use existing modules
    - understand the system in pieces
  - Abstraction
    - allows work in isolation because interfaces guarantee that components will work together

Set 6 8

## Some More Principles

- Anticipation of change
  - allows changes to be absorbed seamlessly
- What makes a good module?
  - High cohesion: all internal parts are closely related.
  - Low coupling: modules rely on each other as little as possible
  - Each module hides its internal structure.
  - Generality allows components to be reused throughout the system
  - Incrementality allows the software to be developed with intermediate working results
- Remember to document your rationale!

Set 6 9

## UML Concepts

- Display the boundary of a system & its major functions using use cases and actors
- Illustrate use case realizations with interaction diagrams
- Illustrate scenarios with use case diagrams and sequence diagrams
- Represent a static structure of a system using class diagrams
- Model the behavior of objects with state transition diagrams
- Reveal the physical implementation architecture with component & deployment diagrams

Set 6 10

## Diagrams in UML

- A diagram is a view into a model
  - Presented from the aspect of a particular stakeholder
  - Provides a partial representation of the system
  - Is (should be?) semantically consistent with other views

Set 6 11

## Types of UML Diagrams

<u>Structure</u>	<u>Behavior</u>
(6 types)	(4 types)
Class diagrams	Activity diagram
Object diagram	Use Case diagram
Package diagram	State machine diagram
Composite structure diagram	Interaction diagrams
Component diagram	Sequence diagram
Deployment Diagram	Communication diagram
	Interaction overview diagram
	Timing diagram

If the appropriate diagram is not part of UML  
*use it anyways*

Set 6 12

## UML & the S/W Process (Design)

- **Use Cases**
    - Define the system Boundaries
  - **Class Diagrams**
    - From a software perspective
      - Show classes & how they interrelate
  - **Sequence Diagrams**
    - For Common Scenarios
      - Pick most significant scenarios from Use Cases
      - Use CRC cards or sequence diagrams to determine how the software should behave
        - Class, Responsibilities, Collaborators (CRC) cards are index cards used to represent
          - the responsibilities of classes
          - interaction between the classes
- **Package Diagrams**
  - Show large-scale organization of the system
- **State Diagrams**
  - Used for classes with complex lifecycles
- **Deployment Diagrams**
  - Show the physical layout of the software

All of these can be used for design

Set 6

13