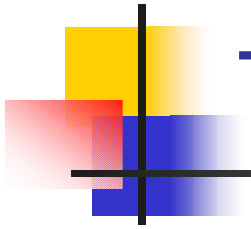
ICS 52
Introduction to Software Engineering

Lecture Notes for Spring Quarter, 2001

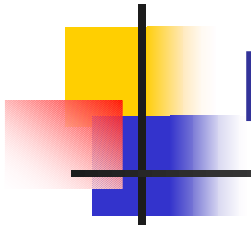
André van der Hoek

Lecture 2-2



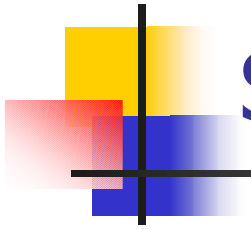
Today's Lecture

- Contents of a requirements specification
- Example requirements session
- Acceptance test plan



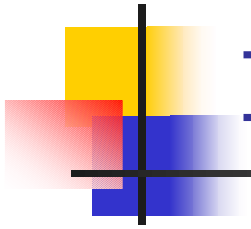
Requirements Specification

- Serves as the fundamental reference point between customer and software producer
- Defines capabilities to be provided without saying how they should be provided
  - Defines the "what"
  - Does not define the "how"
- Defines environmental requirements on the software to guide the implementers
  - Platforms
  - Implementation language(s)
- Defines software qualities



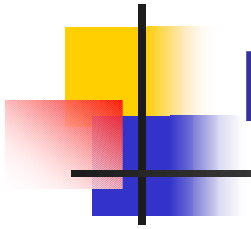
Structure

- Introduction
- Executive summary
- Application context
- Functional requirements
- Environmental requirements
- Software qualities
- Other requirements
- Time schedule
- Potential risks
- Future changes
- Glossary
- Reference documents



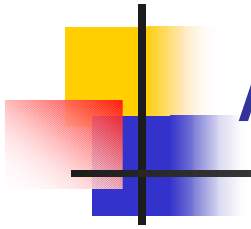
Introduction

- What is this document about?
- Who was it created for?
- Who created it?
- Outline



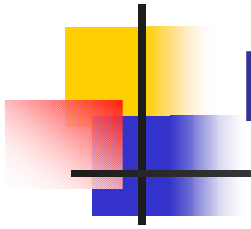
Executive Summary

- Short, succinct, concise, to-the-point, description
  - Usually no more than one page
- Identifies main goals
- Identifies key features
- Identifies key risks/obstacles



Application Context

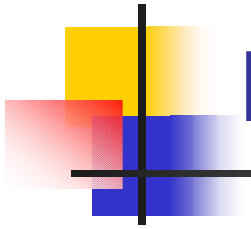
- Describes the situation in which the software will be used
  - How will the situation change as a result of introducing the software?
  - "World Model"
- Identifies all things that the system affects
  - Objects, processes, other software, hardware, and people
  - Provides an abstraction for each of those, characterizing the properties and behaviors that are relevant to the software system
- Identifies fundamental assumptions



Functional Requirements

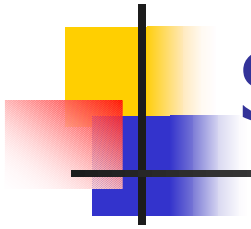
- Identifies all concepts, functions, features, and information that the system provides to its users
- Provides an abstraction for each of those, characterizing the properties and functions that are relevant to the user
  - What is the system supposed to do?
  - What information does the system need?
  - What is supposed to happen when something goes wrong?

An approximate user interface is part of functional requirements



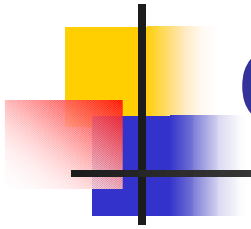
Environmental Requirements

- Platforms
  - Hardware
    - Operating systems, types of machines, memory size, hard disk space
  - Software
    - CORBA, Jini, DCOM, 4GL, …
- Programming language(s)
- Standards



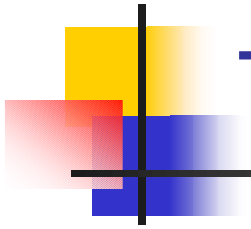
Software Qualities

- Correctness
- Reliability
- Robustness
- Performance
- User friendliness
- Verifiability
- Maintainability
- Repairability
- Safety
- Evolvability
- Reusability
- Portability
- Understandability
- Interoperability
- Productivity
- Size
- Timeliness
- Visibility



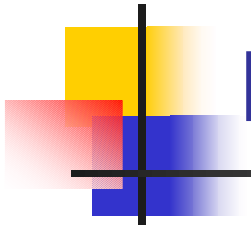
Other Requirements

- What about cost?
- What about documentation?
- What about manuals?
- What about tutorials?
- What about on-the-job training?
- What about requirements that do not fit in any of the previous categories?



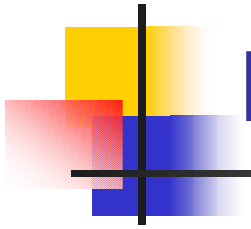
Time Schedule

- By when should all of this be done?
  - Initial delivery date
  - Acceptance period
  - Final delivery date
- What are some important milestones to be reached?
  - Architectural design completed
  - Module design completed
  - Implementation completed
  - Testing completed



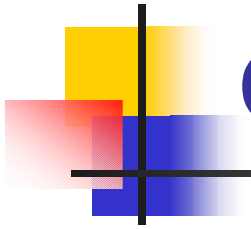
Potential Risks

- Any project faces risks
  - Boehm's top ten risks (see lecture 1.2)
  - It is important to identify those risks *up-front* so the customer *and you (!)* are aware of them
    - One of the requirements could be to explicitly address the risks



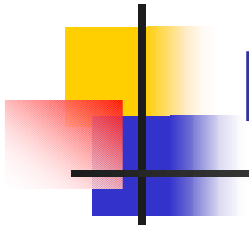
Future Changes

- Any project faces changes over time
  - It is important to identify those changes *up-front* so the customer *and you (!)* are aware of them
  - These changes could simply pertain to potential future enhancements to the product
    - One of the requirements could be to build the product such that it can accommodate future changes



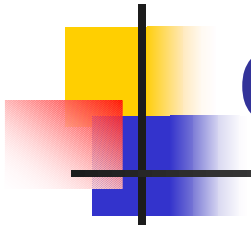
Glossary

- Precise definitions of terms used throughout the requirements document



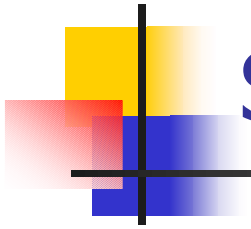
Reference Documents

- Pointers to existing processes and tools used within an organization
- Pointers to other, existing software that provide similar functionality
- Pointers to literature



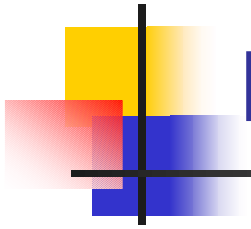
Observations

- Document is structured to address the fundamental principles
  - Rigor
  - Separation of concerns
    - Modularity
    - Abstraction
  - Anticipation of change
  - Generality
  - Incrementality
- Not every project requires every section of the document



Specification Methods

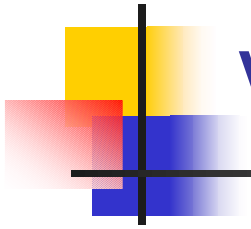
- Natural language
- Data flow diagrams
  - Office automation
- Finite state machines
  - Telephone systems
  - Coin-operated machines
- Petri nets
  - Production plants
- Formulas
  - Matrix inversion package



Helpful Techniques

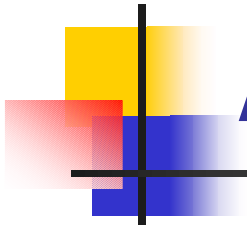
- Functional approach
  - List of features
  - Input and output
  - "Recipe"
- World model approach
  - List of objects
  - Attributes and methods
  - "Ingredients and their possible uses"

Both lead to a "shopping list" and "dinner"



Verification

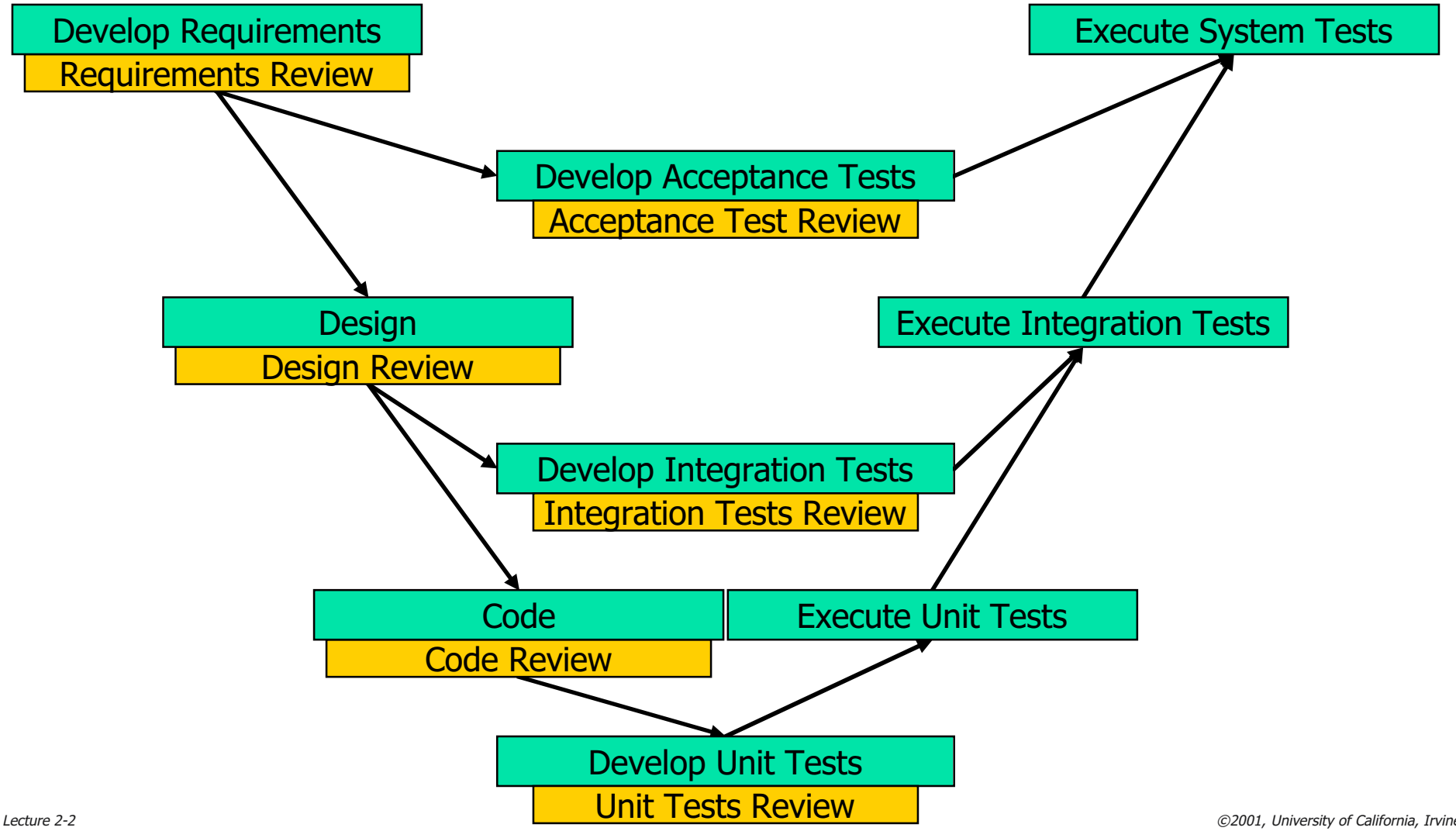
- Is the requirements specification *complete?*
- Is each of the requirements *understandable?*
- Is each of the requirements *unambiguous?*
- Are any of the requirements *in conflict?*
- Can each of the requirements be *verified?*
- Are are all terms and concepts *defined?*

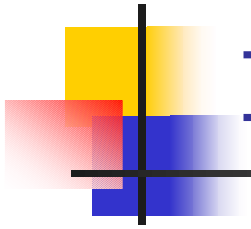


Acceptance Test Plan

- Accompanies a requirements specification
- Specifies, in an operational way, consistency between the requirements specification and the system that will be delivered
- Binds a customer to accept the delivered system if it passes all the tests
- Covers all aspects of the requirements specification

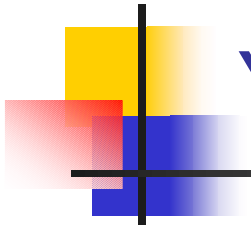
V-Model of Development and Testing





In-Class Example

"*The french fries with mayonnaise place*"



Your Tasks

- Read and study slides of this lecture
- Read Chapter 5 of Ghezzi, Jazayeri, and Mandrioli
  - Be familiar with some of the more formal requirements analysis techniques
  - This is a *big* chapter