Embedded Systems Architecture

UC Irvine Extension

EECS X497.36 (3 units)

Who am I?

- My name is Clyde R. Visser, P.E.
- I have a BSEE degree
- I have over 25 years experience in telecommunication, data communication, medical, power conversion and motor drive systems industries using embedded systems

How can you contact me?

- Email: CVisser@UCI.edu
- Forums (fora?)

Introduction Message

- Please post a welcome message to the Welcome Messages forum. Include:
  - Level of education
  - Level of experience
  - What are your expectations for this course

Course Description

- Learn about the architecture of embedded systems and explore the differences between embedded designs and traditional electronic device design. The special demands on embedded systems including portability, low power usage, and miniaturisation dictate a different approach. This course introduces models and architectures, and covers such topics as specification, system partitioning, design quality, and developing synthesizable models.

Prerequisites

- Eecs 805, C Programming for Embedded Systems
- Eecs X497.32, Fundamentals of Embedded Systems Design and Programming
- or equivalent experience or education

Learning Objectives

- Understand the process of designing embedded systems
- Understand the architecture of 3 representative CPUs
- Understand the hardware architecture of embedded systems

Learning Objectives (continued)

- Understand the software architecture of embedded systems
- Be able to select appropriate hardware modules
- Make software/hardware design decisions to optimize performance/cost

This is a required course for the Embedded Systems Engineering Certificate.

Methodology

- Lecture
  - Survey of technologies used in hardware and software architectures design and development.
- Lab
  - Study and use of structured methodology for design project requirements & architecture specification.

Grading Scale

A = 93 - 100		A- = 90 - 92
B+ = 86 – 89	B = 83 – 85	B- = 80 - 82
C+ = 76 – 79	C = 73 – 75	C- = 70 - 72
D+ = 66 – 69	D = 63 – 65	D- = 60 - 62
F= 0 - 59		

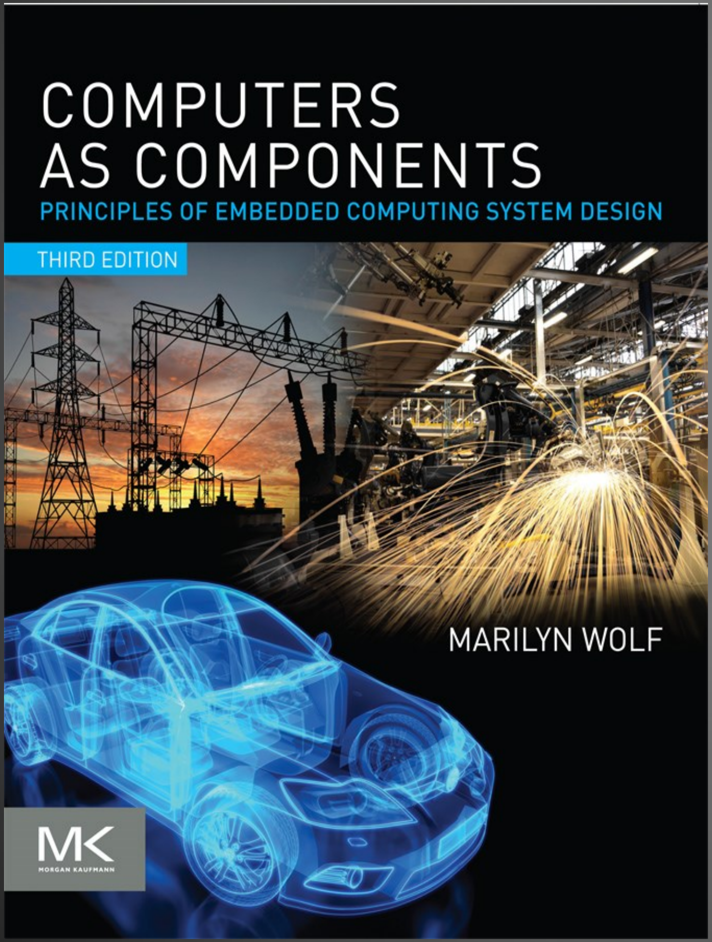
Quizzes

- One quiz per session.
- Quizzes will cover assigned reading, lecture, and lab material.

Class Meeting Information

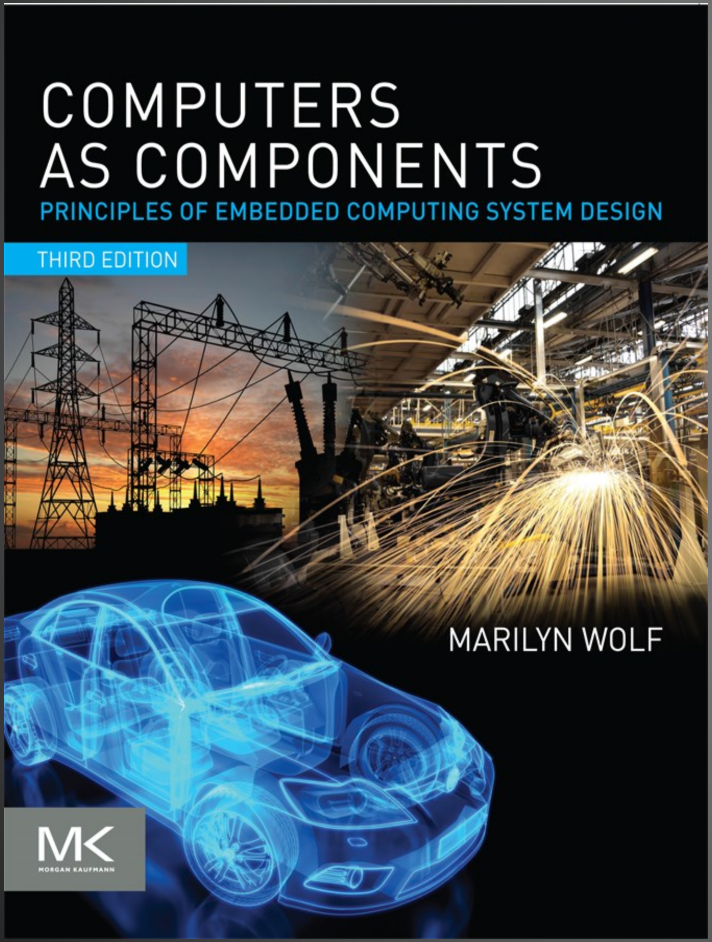
- Online
- 11 class meetings

Course Text



- Computers as Components, Third Edition: Principles of Embedded Computing System Design (The Morgan Kaufmann Series in Computer Architecture and Design) (ISBN-13: 978-0123884367)

Course Text



- Additional materials written by the author can be found online
  - http://www.marilynwolf.us/

Evaluation of Student Performance

Quizzes (9 @ 5% each)	45%
Lecture Assignments (2 @ 10% each)	20%
Lab Assignments (1 @ 20%, 1 @ 10%)	30%
Introduction	5%

Quizzes

- One quiz per session.
- Quizzes will be given at the end of each session.
- Quizzes will cover assigned reading, lecture, and lab material.

Lecture Assignments

- Two (2) Research Papers due through out the 10 week course. A third can be done for extra credit.
- 2 to 4 pages of original material.
- Chose research topics related to previous or current lecture material of due date.
- Include attributions and references, if any.
- *Unless otherwise requested, I reserve the right to use your research papers for future classes as reference material.*

Lab Assignments

- Design Project
  - Requirements model specification.
  - Architecture model specification.

Class Participation

- Post questions.
- Make observations.
- Point out errors when you see them.

Words of Wisdom

- Years ago, these words of wisdom could be found printed on the side of a soda bottle:
  - *NO DEPOSIT, NO RETURN*

Course Outline

- Class 1 Lecture: Introductions & course description.
- Class 1 Lab: Lab Introduction.

- Class 2 Lecture: Understanding embedded systems basics of design & implementation.
- Class 2 Lab: Data Flow Diagrams (DFD).

- Class 3 Lecture: Cover Processor Basics, ARM, SHARC, and AVR CPU architectures.
- Class 3 Lab: Control Flow Diagrams (CFD).

Course Outline (continued)

- Class 4 Lecture: CPU architecture basics.
- Class 4 Lab: Requirements Model Overview.

- Class 5 Lecture: The Embedded Computing Platform.
- Class 5 Lab: Requirements Model Presentation.

- Class 6 Lecture: SoC: System-on-a-Chip
- Class 6 Lab: Architecture Model #1.

- Class 7 Lecture: Software Development Tools.
- Class 7 Lab: Architecture Model #2.

Course Outline (continued)

- Class 8 Lecture: Multitasking and operating system basics.
- Class 8 Lab: Architecture Model Partitioning Considerations.

- Class 9 Lecture: Hardware Accelerators.
- Class 9 Lab: Architecture Model Presentation

- Class 10 Lecture: Communications & Networks.
- Class 10 Lab: Design Project Presentations.

An Apology

- I will normally keep the Power Point slide rate to 10 slides or less per hour. *This next hour is the exception!*
- This material comes primarily from chapter 7.

System Design Techniques

- Design Methodologies
- Requirements Analysis
- Specification

Design Methodologies

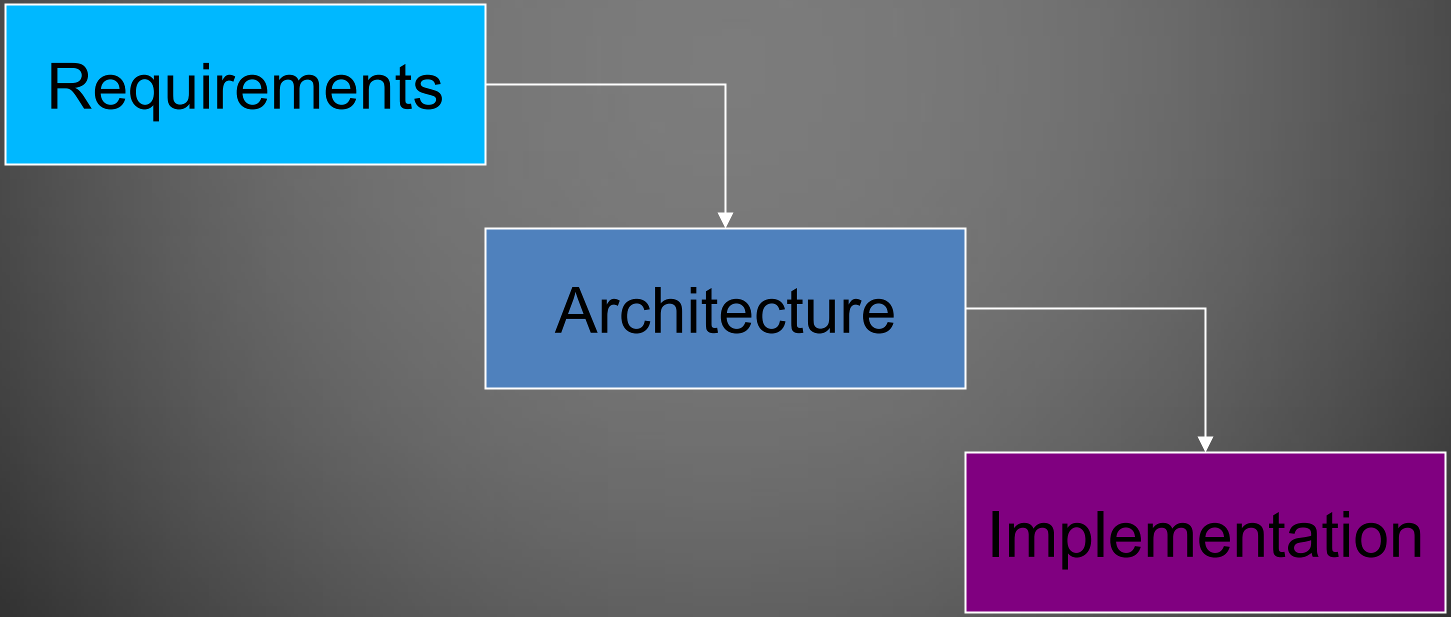
- Process for creating a system.
- Many systems are complex:
  - large specifications;
  - multiple designers;
  - interface to manufacturing.
- Proper processes improve:
  - quality;
  - cost of design and manufacture.

Design Flow

- Design flow: sequence of steps in a design methodology.
- May be partially or fully automated.
  - Use tools to transform, verify design.
- Design flow is one component of methodology. Methodology also includes management organization, etc.

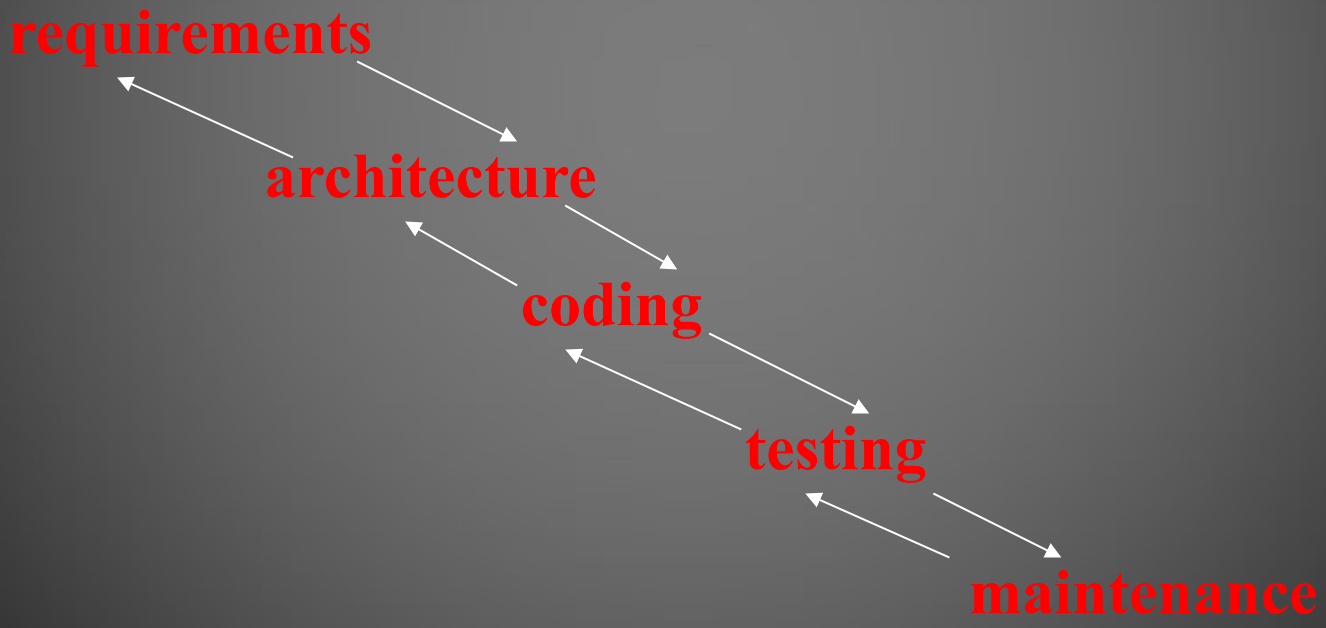
Waterfall Model

- Early model for software development:



Waterfall Model

- Early model for software development:



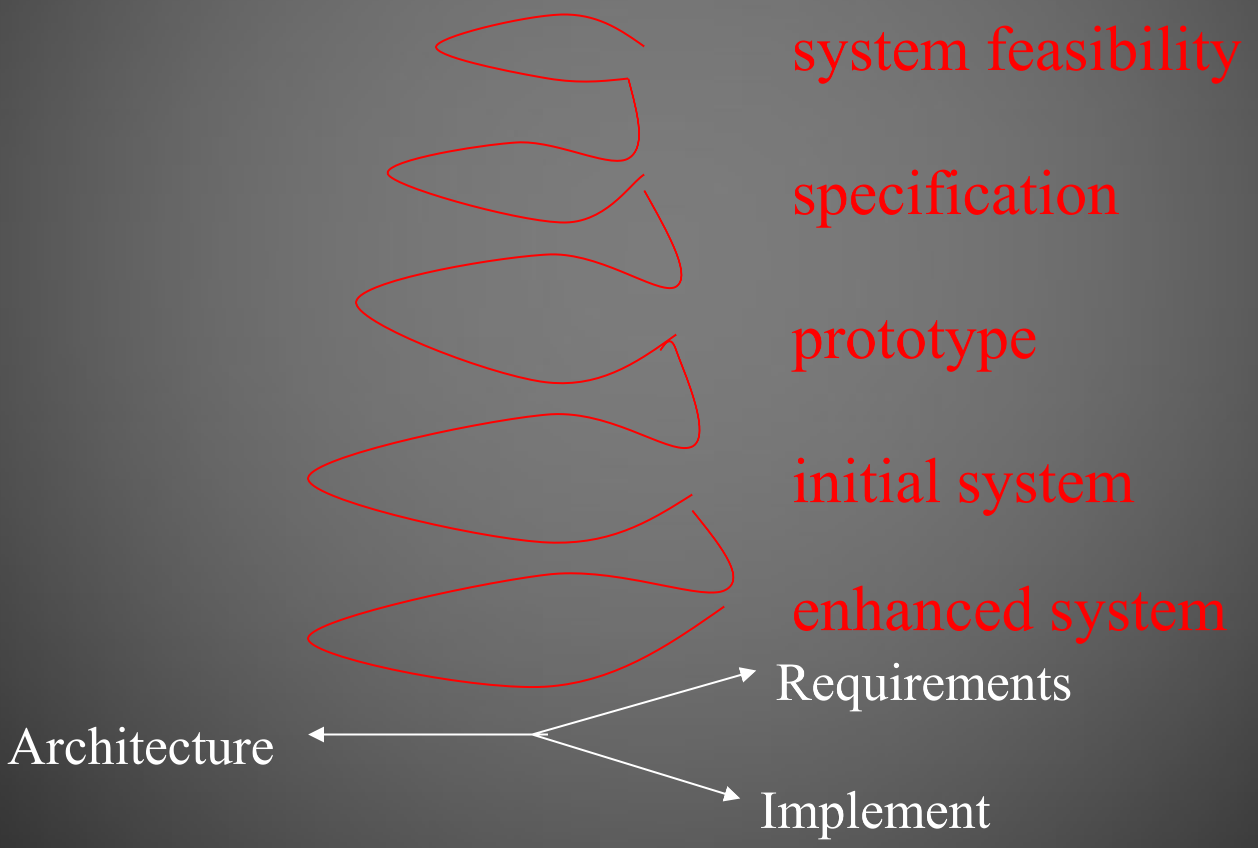
Waterfall Model Steps

- Requirements: determine basic characteristics.
- Architecture: decompose into basic modules.
- Coding: implement and integrate.
- Testing: exercise and uncover bugs.
- Maintenance: deploy, fix bugs, upgrade.

Waterfall Model Critique

- Only local feedback---may need iterations between coding and requirements, for example.
- Doesn’t integrate top-down and bottom-up design.
- Assumes hardware is given.

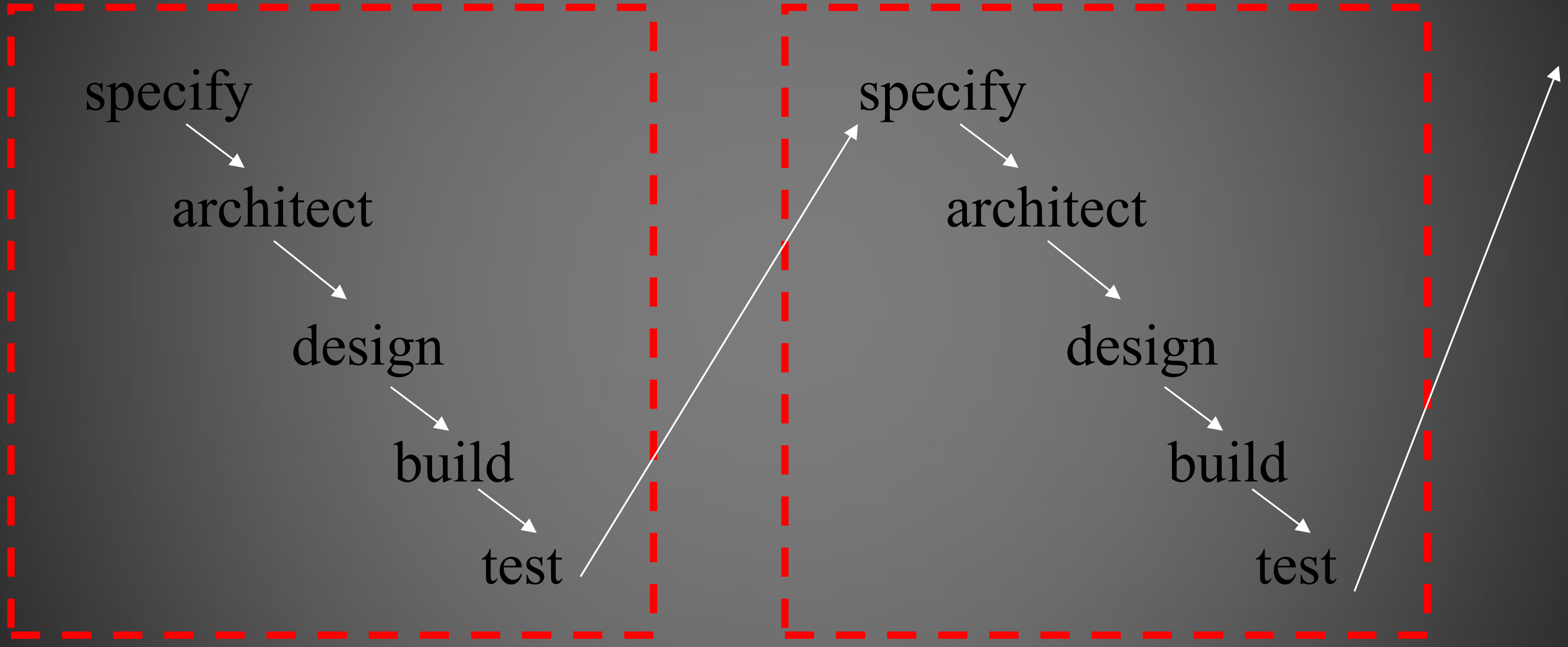
Spiral Model



Spiral Model Critique

- Successive refinement of system.
  - Start with mock-ups, move through simple systems to full-scale systems.
- Provides bottom-up feedback from previous stages.
- Working through stages may take too much time.

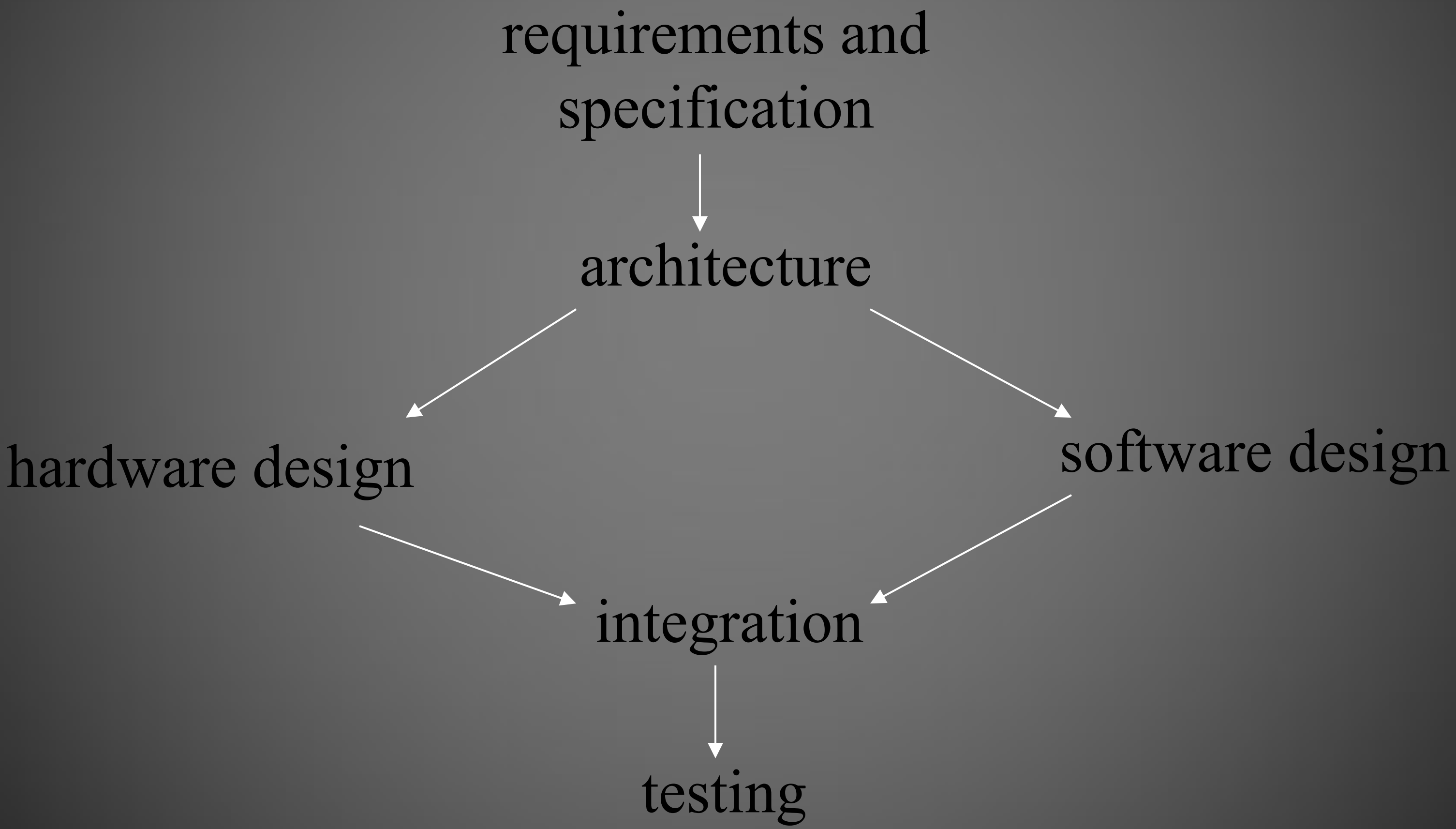
Successive Refinement Model



initial system

refined system

Hardware/Software Design Flow



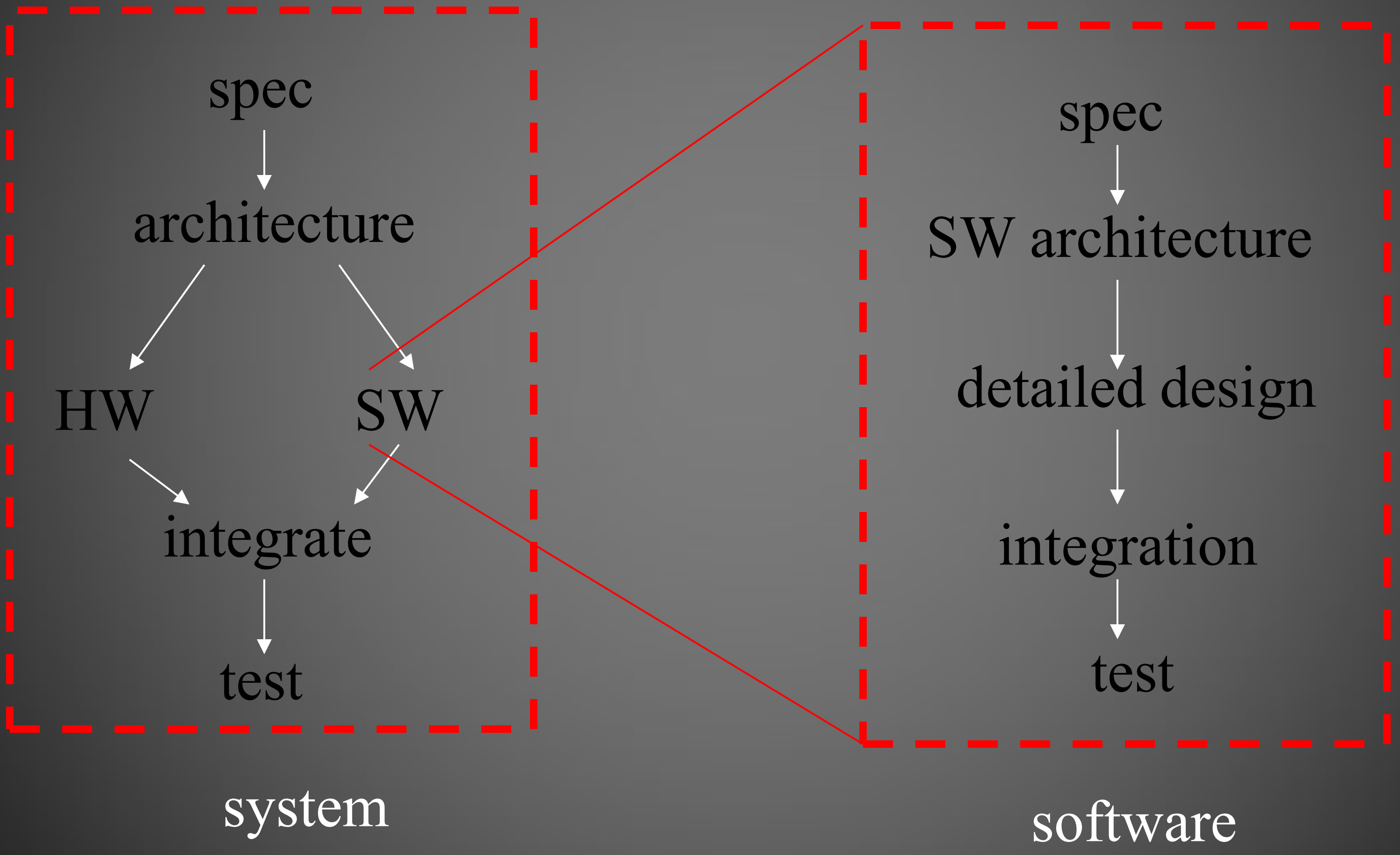
Co-design Methodology

- Must architect hardware and software together:
  - provide sufficient resources;
  - avoid software bottlenecks.
- Can build pieces somewhat independently, but integration is major step.
- Also requires bottom-up feedback.

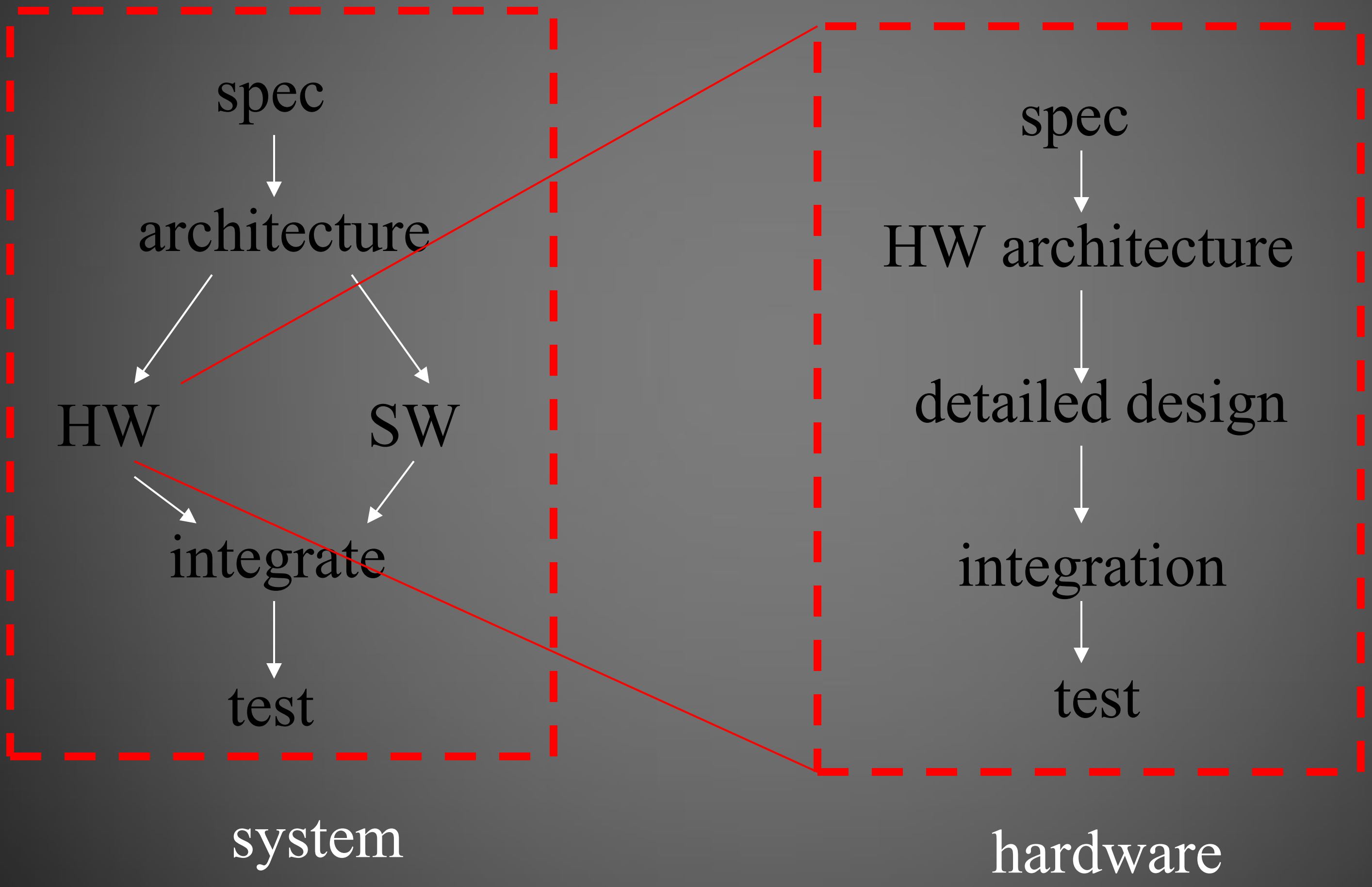
Hierarchical Design Flow

- Embedded systems must be designed across multiple levels of abstraction:
  - system architecture;
  - hardware and software systems;
  - hardware and software components.
- Often need design flows within design flows.

Hierarchical HW/SW Flow



Hierarchical HW/SW Flow



Concurrent Engineering

- Large projects use many people from multiple disciplines.
- Work on several tasks at once to reduce design time.
- Feedback between tasks helps improve quality, reduce number of later design problems.

Concurrent Engineering Techniques

- Cross-functional teams.
- Concurrent product realization.
- Incremental information sharing.
- Integrated product management.
- Supplier involvement.
- Customer focus.

Requirements Analysis

- Requirements: informal description of what customer wants.
- Specification: precise description of what design team should deliver.
- Requirements phase links customers with designers.

Types of Requirements

- Functional: input/output relationships.
- Non-functional:
  - timing;
  - power consumption;
  - manufacturing cost;
  - physical size;
  - time-to-market;
  - reliability.

Good Requirements

- Correct.
- Unambiguous.
- Complete.
- Verifiable: is each requirement satisfied in the final system?
- Consistent: requirements do not contradict each other.

Good Requirements, cont'd.

- Modifiable: can update requirements easily.
- Traceable:
  - know why each requirement exists;
  - go from source documents to requirements;
  - go from requirement to implementation;
  - back from implementation to requirement.

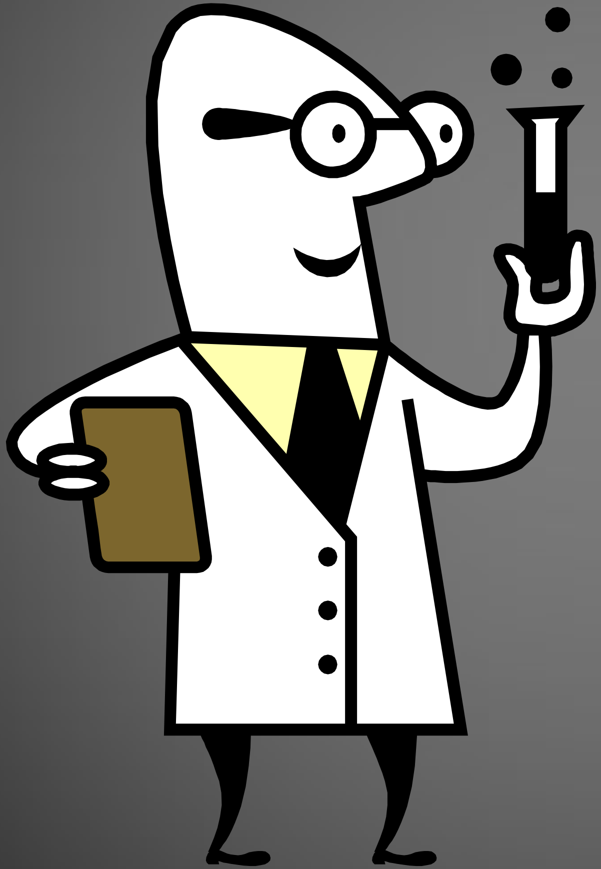
Setting Requirements

- Customer interviews.
- Comparison with competitors.
- Sales feedback.
- Mock-ups, prototypes.
- Next-bench syndrome (HP): design a product for someone like you.

Specifications

- Capture functional and non-functional properties:
  - verify correctness of spec;
  - compare spec to implementation.
- Many specification styles:
  - control-oriented Vs. data-oriented;
  - textual Vs. graphical.
- UML is one specification/design language.

Lab Session #1



- Lab Introduction

Lab Introduction

- The lab consists of doing a paper only Design Project.
- Choice of the project is up to the student.
- *Nothing proprietary, please!*

Lab Introduction (continued)

- We’re going to use the Hatley Pirbhai Methodology (HPM) of describing (through graphical means) a requirements model and an architecture model of the system as described in *Strategies for Real-Time System Specification*.

Lab Introduction (continued)

- The design itself must contain one or more embedded processors and be of sufficient complexity to *warrant* the use of microprocessor technology.

Lab Introduction (continued)

- The methodology builds on the structured methods developed by Edward Yourdon, Tom DeMarco, et al for the requirements model and adds an architecture model which links back to the requirements.