

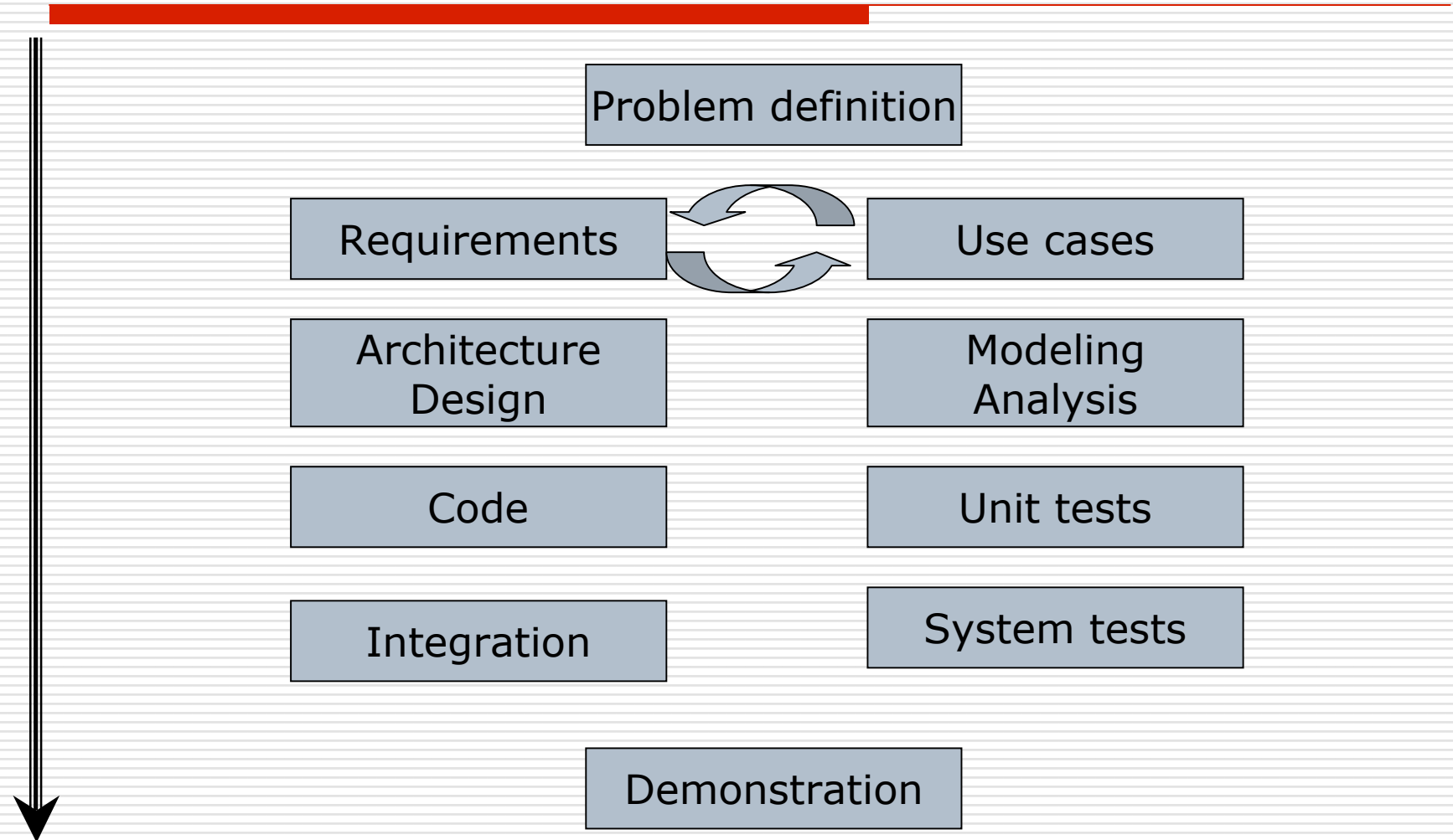
# COMP314:

## Requirements and Use Cases

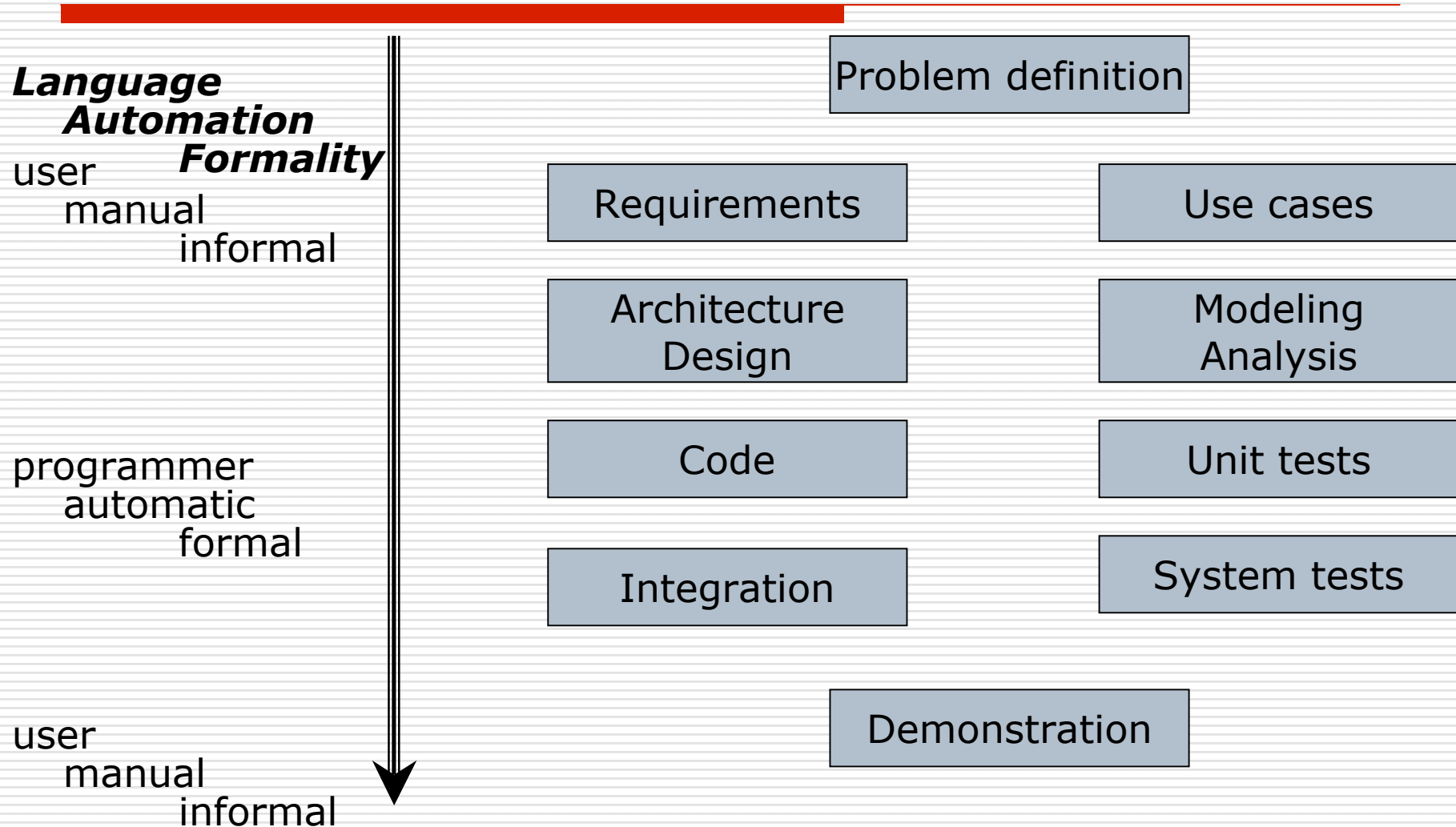
---

Chapter 3 of "Code Complete"

# Intermediate size project



# Intermediate size project



# Prerequisites

---

Problem definition

Requirements

Use cases

Architecture  
Design

Modeling  
Analysis

Code

Unit tests

Integration

System tests

Demonstration

# Project Iterations

---

Quiet point

a system that can be demonstrated to the user

Following our principles above

make each iteration as short as possible  
in this case weeks of work

Think about the iteration beforehand

you may be surprised how few features are needed

Where is the uncertainty?

what the user wants? - do the technically easy stuff  
some technical problem? - do the critical technical stuff

# Why do Prerequisites Work?

---

According to our principles we shouldn't do wasted work.

Will constructing a prerequisite give information we didn't know before?

Are there risks associated with deciding what we are going to do before doing it?

# Why do Prerequisites Work?

---

The answer will depend on the size and nature of the project

University assignment - probably not  
given problem already and it is small

Windows operating system - you bet

Wicked problems -  
you don't know what the problem is until you have solved it  
write a prototype and throw it away  
(better let your boss know that this one is expensive and risky)

# Why do Prerequisites Work?

---

## Self-fulfilling statements

We'd better start coding right away because we are going to have a lot of debugging to do

We haven't planned much time for testing because we're not going to find many defects

We've investigated requirements and design so much that I can't think of any major problems we'll run into during coding or debugging

# Cost of fixing defects

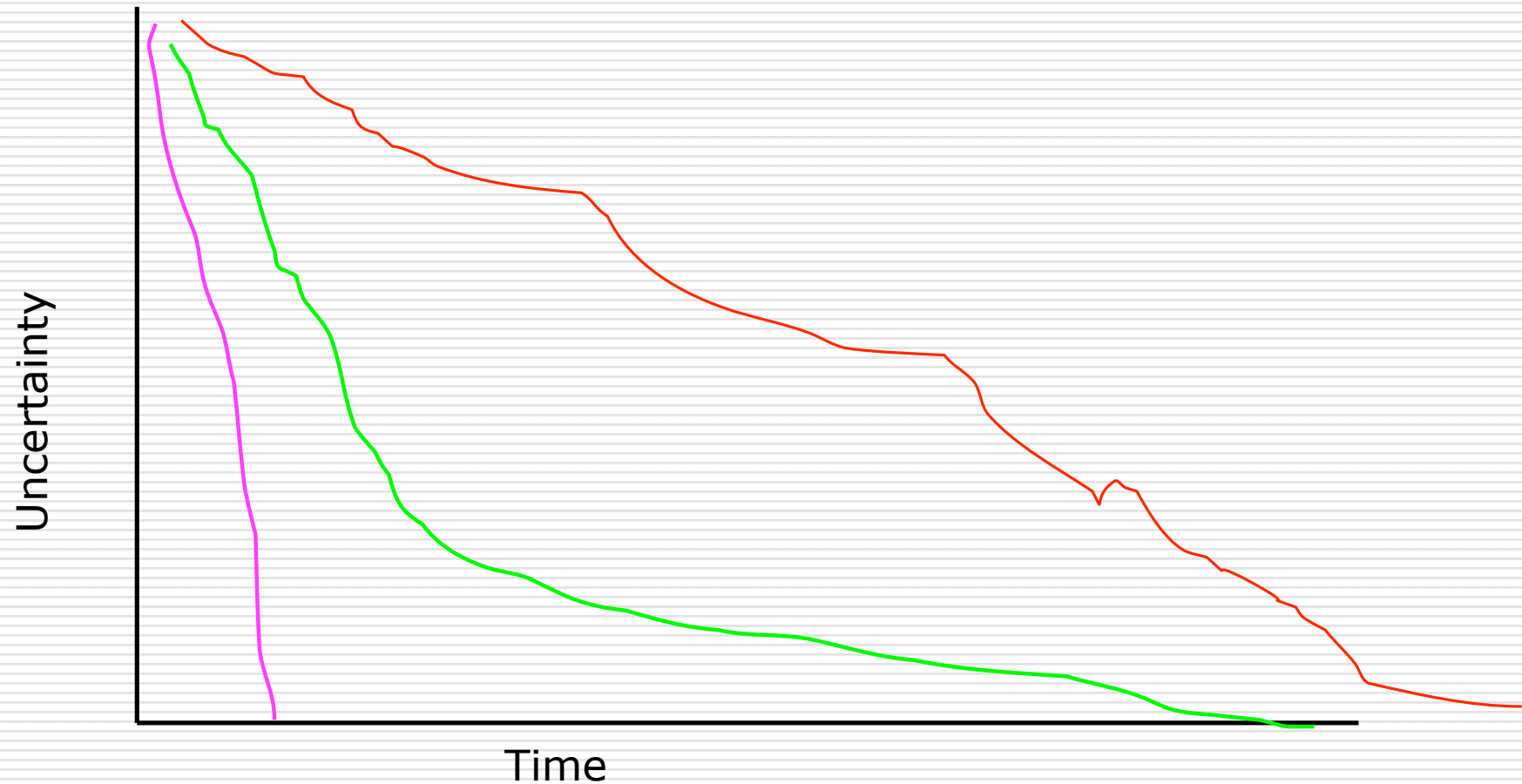
---

Table 3-1 Code complete

Introduced	Requirements	Design	Code	Detected	Integration	Post-release
Requirements		1	3	5-10	10	10-100
Design			1	10	15	25-100
Code				1	10	10-25

# Uncertainty against time

---



# Project Deliverables

---

(see page on website)

- Minutes of meetings (attendance)
- Planning
- Requirements
- Design and architecture
- Code (documented)
- Unit Testing
- User documentation
- System Testing
- Demonstration
- Personal Summary

# Problem definition

---

A good problem definition:  
in the user's language  
does NOT specify the solution

# Examples of problem definitions

---

- ❑ The genomics search program is not robust when dealing with large input sets
- ❑ Need to know the total size of genomic data before splitting it into parts
- ❑ Need a market leading operating system.
- ❑ Be able to recognize handwritten text (within 15 man weeks)

# Where have we got to?

---

Problem definition

Requirements

Use cases

Architecture  
Design

Modeling  
Analysis

Code

Unit tests

Integration

System tests

Demonstration

# Common to both Use Cases and Requirements

---

- Are all the tasks the user wants to perform specified?
- Written in the user's language?
- Avoid specifying the design?
- At a consistent level of detail?
- Is each item relevant to the problem and its solution?

# Common to both Use Cases and Requirements - continued

---

- Clear enough to be turned over to an independent group for construction (including tests) and still be understood?
- Are you comfortable with every item? Have you eliminated items that are impossible to implement and included just to appease your customer or your boss?

# Use Cases

---

- Each specification is covered by a use case?

# Requirements - functional

---

- Are all the inputs to the system specified including their source, accuracy, range of values and frequency.
- Are all the outputs from the system specified, including their destination, accuracy, range of values and frequency.
- Are all output formats specified for reports and so on?

# Requirements - functional - cont.

---

- Are all the external hardware and software interfaces and external communication interfaces specified, including handshaking, error-checking, and communications protocols?
- Is the data used in each task and the data resulting from each task specified?
- Is the level of security specified?

# Requirements - non-functional

---

- Is the expected response time, from the users point of view, specified for all necessary operations?
- Is the reliability specified, including the consequences of software failure, the vital information that needs to be protected from failure, and the strategy for error detection and recovery?
- Are minimum machine memory and free disk space specified?
- Is the maintainability of the system specified, including its ability to adapt to changes in specific functionality, changes in the operating environment, and changes in its interfaces with other software?

# Requirements - general

---

- Does each requirement avoid conflicts with other requirements?
- Are acceptable tradeoffs between competing attributes specified?
- Is each requirement testable?
- Are possible changes to the requirements specified, including the likelihood of each change?
- Are areas of incompleteness specified?

# Requirements - general - cont.

---

- Are the requirements complete in the sense that if the product satisfies every requirement it will be acceptable?
- Each specification is short (no more than a sentence or two)?
- Each requirement is numbered for reference?