CSE 331
Software Design & Implementation

Kevin Zatloukal

Fall 2017

Requirements Analysis

Software Lifecycle

Idealized procedure for developing software (program or **module**):

1. Requirements analysis
   - define what it means to solve the problem
2. Design
   - make a blueprint for the solution
3. Implementation
   - write the code
4. Testing
   - find and remove bugs

(Rarely proceeds smoothly from one phase to the next.)

Requirements Analysis

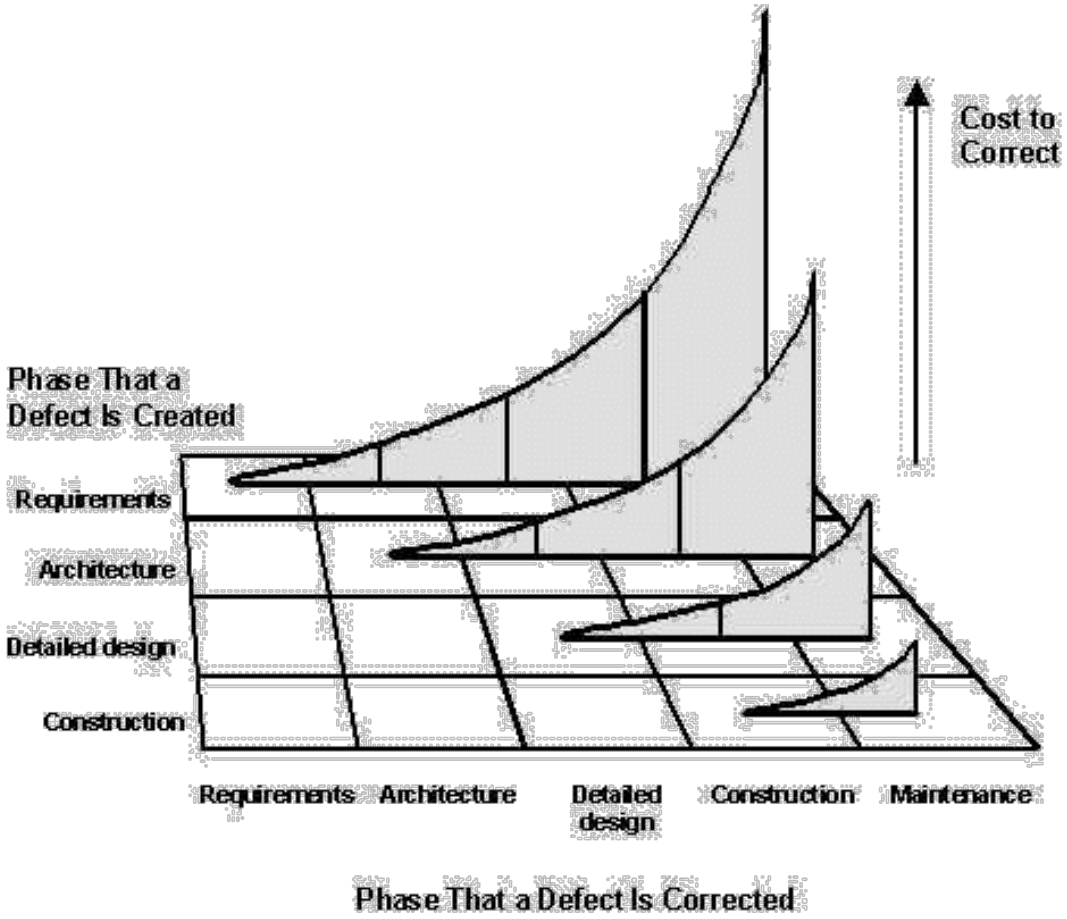
- Find conditions that must be met by a solution
  - define what it means to be a “solution”
  - capture the needs of the user / client
- Come from discussion with customer and further analysis

For most software projects, requirements analysis is both

- the most important phase
- the most difficult phase

Why is requirements analysis so important?

Cost to Fix a Bug



From Steve McConnell's article:
http://www.stevemcconnell.com/articles/art04.htm

Cost to Fix a Bug in Requirements

Suppose you discover that you missed a requirement…

It might be easy to fix OR

It might require throwing your code away and starting over

- e.g., if you discover the data structure needs O(1) updates
  - you'll have to throw away your Red-Black tree code
  - start over writing a hash table
- e.g., if you discover you needs `findNextLargest` method
  - you'll have to throw away your hash table
  - start over writing a Red-Black tree

Requirements Analysis

- Applies to both:
  - designing a whole program
  - designing an individual module

- For a program, requirements come from users
- For a module, requirements come from clients
  - clients are authors of other modules that want to call yours

Use Case Analysis

Simple and common approach to requirements analysis

- see CSE 403 & 440/441 for much more…

Put together a collection of **use cases**, each of which

- describes the goal of the user/client in a specific scenario
- describes the interaction of the user/client with the software
  - each input and response
- from the user/client's point of view

Deduce requirements from the use cases

Example: Library Web Site

Use Case 1:

1. user looks up book
2. user finds that book is available at Odegaard

Use Case 2:

1. user looks up book
2. user finds that book is checked out from Odegaard
3. user places hold on book
4. (later) user gets message that book is on hold shelf

Requirements:

- ability to search for books, place holds, view current holds, …
- user accounts have an associated email address or phone #

HW5 – Graph ADT

In HW5, you are not given a specification to implement

- only given an *incomplete* set of requirements
- you have to **design**, **implement**, and **test** your solution

You should complete the requirements analysis

- any time spent on this can pay for itself many times over
- **look at HW6** (use case!) to determine necessary operations
  - see how the graph class will be used
- add more operations *only* if you have strong reason to do so
  - e.g., containers should let clients retrieve anything added
    - clients should not have to remember what they added themselves
  - best way to i**mprove productivity** is to *write less code*