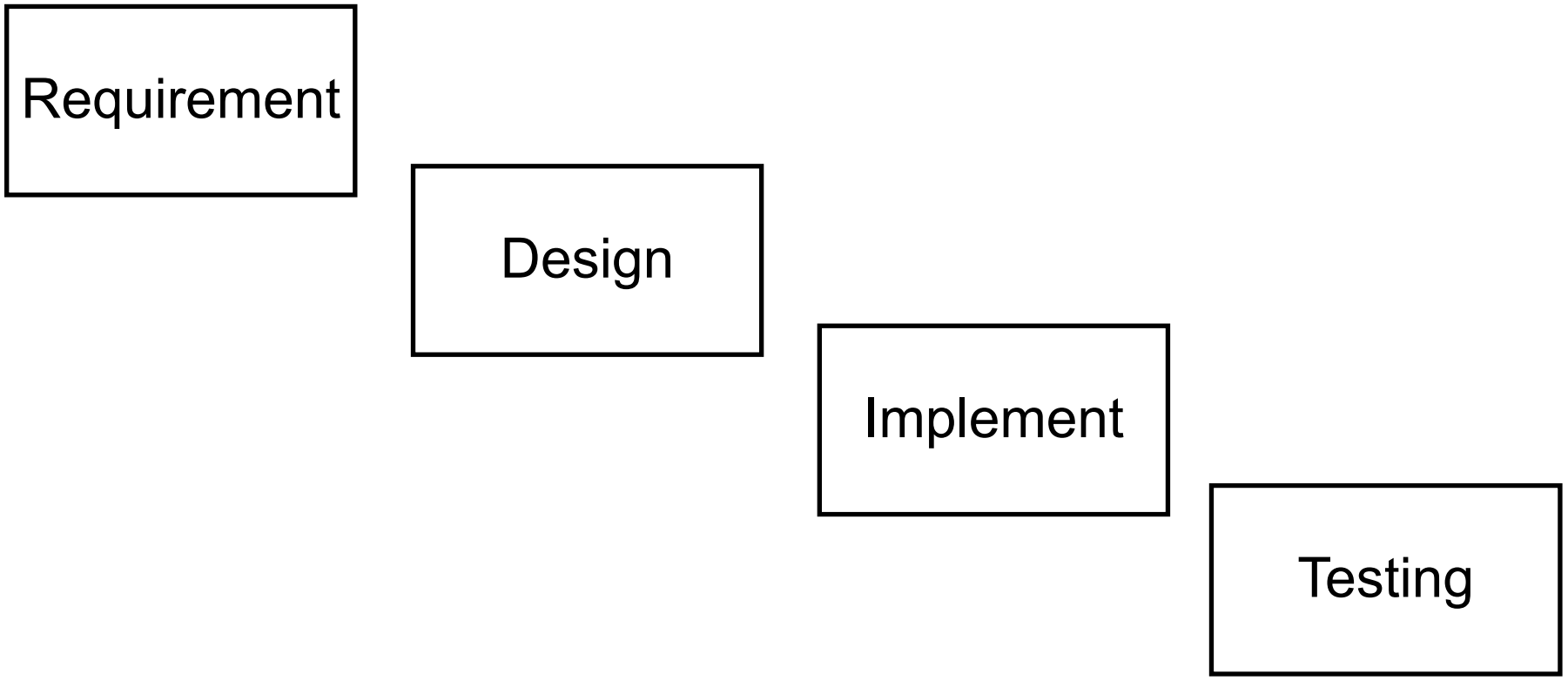
Clement Su

Design Document Introduction

Development Cycle (WaterFall)



Why Design Document?

- Communicate with
  - Architect
  - Peer developers
  - Tester
  - Document team
  - Successors
  - Yourself
- Help developer/architect to think more
- Reduce possibility of rework

Different Design Document

- High level Design Document
  - For architect (or written by architect)
  - Focus on system level design
- Implement level Design Document
  - For peer developer (or whoever want to know detail)
  - Focus on component level implementation detail
- Both are important and valuable

Keys of a Good Design Document

- Showing that the requirement is fulfilled
- Describe the design clearly (with Diagram, UML, etc)
- Reveal the reason (benefit) of choosing this design
- List assumptions, risks, issues and future extension

Components of a Design Document

- The goal of this implementation
- High level entities
- For each entity, a detail description
  - How to use
  - How to configure
  - UML Model
  - How does it interact with others
- Benefits, assumptions, risks, and other issues

Tips

- Prepare a skeleton, then fill it up.
- Pretend you are the readers, what do you want to see?
- Let others to read and ask questions and improve the content.

What to avoid

- Do not assume readers' background knowledge
- Do not use too many abbreviation or create terminology

Design Document Example – Requirement form Customer

- Our hospital registration system needs to be ported to the application running on mobile devices
- The system should be High Availability

Design Document Example – Requirement after SA

- Server Side
  - Move the infrastructure to cloud
  - Need to be convert into RESTful web service and be available on hospital registration server
- Client Side
  - Develop an Android based hospital registration application (ObjectC is the next target)
  - User can register/login/logout
  - User are Aministrator, Doctor, Patient
  - etc…

Documentation in the code

- Copyright claim
- Javadoc
- ESLint

  The pluggable linting utility for JavaScript and JSX
- Code comments

Javadoc example

```
/**
 * Returns an Image object that can then be painted on the screen.
 * The url argument must specify an absolute {@link URL}. The name
 * argument is a specifier that is relative to the url argument.
 * <p>
 * This method always returns immediately, whether or not the
 * image exists. When this applet attempts to draw the image on
 * the screen, the data will be loaded. The graphics primitives
 * that draw the image will incrementally paint on the screen.
 *
 * @param  url  an absolute URL giving the base location of the image
 * @param  name the location of the image, relative to the url argument
 * @return      the image at the specified URL
 * @see         Image
 */
public Image getImage(URL url, String name) {
      try {
          return getImage(new URL(url, name));
      } catch (MalformedURLException e) {
          return null;
      }
}
```

Agile Development

- lightweight development method
- Key of agile development
  - *Individuals and interactions* over processes and tools
  - *Working software* over comprehensive documentation
  - *Customer collaboration* over contract negotiation
  - *Responding to change* over following a plan

Design Document in Agile

- Documentation should take on a collaborative nature.
- Focus on just barely good enough documentation and avoid big upfront details.
- Documentation can take many forms.