

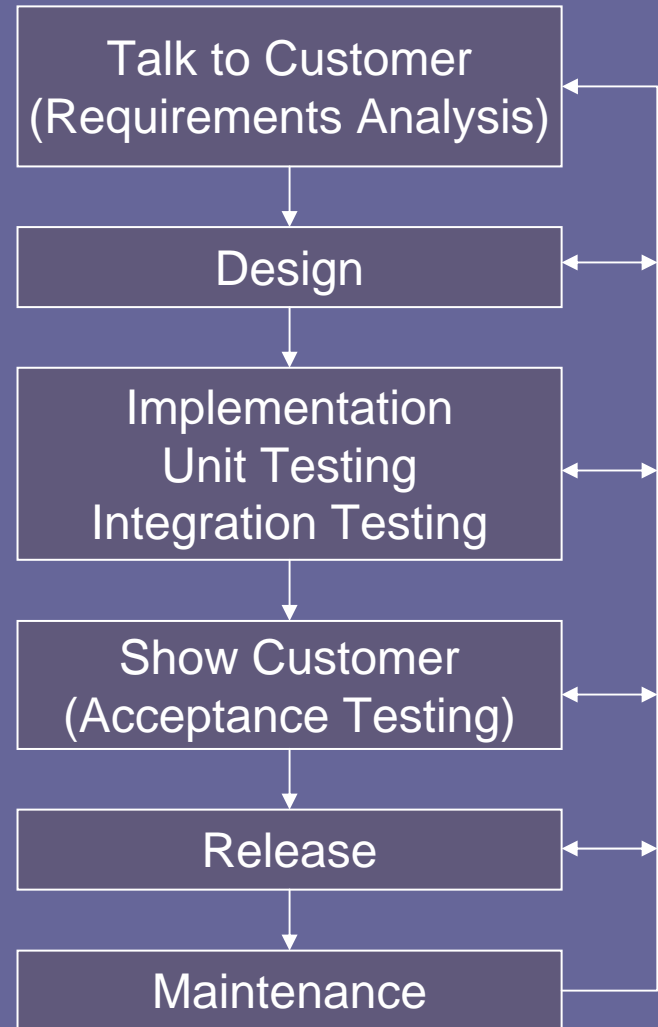
CS 3

Introduction to Software Engineering

11: Software Life Cycle;
Requirements Analysis

Software Development Cycle

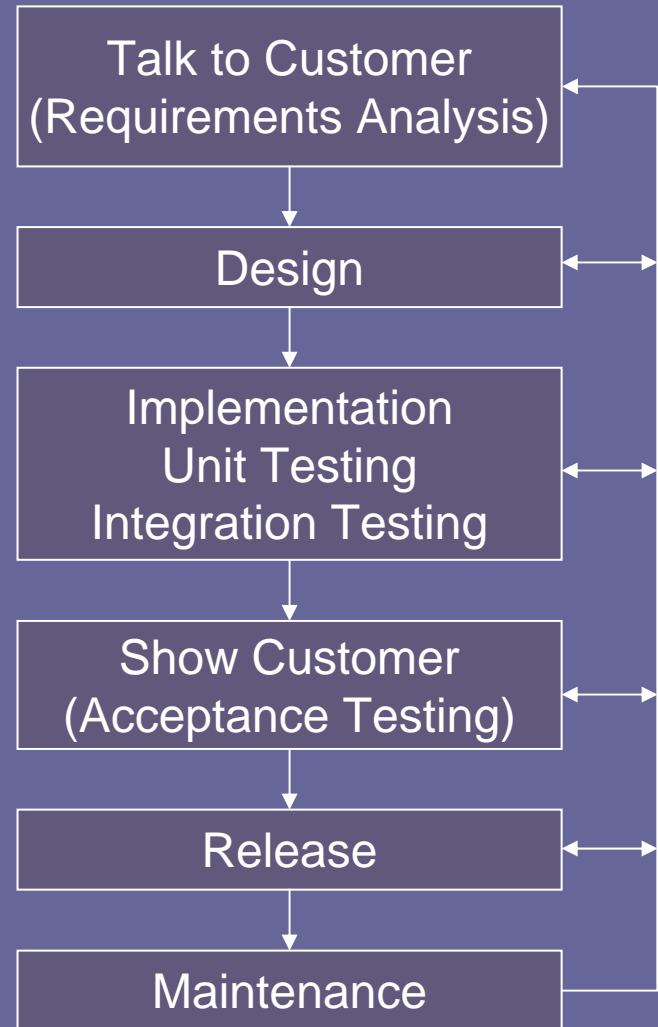
- Interact with customer to learn what's needed.
 - Write it down: Requirements specification.
- Make plan for system.
 - Write it down: Design.
- Implement it.
 - Testing is key to quality.
- When customer's satisfied, done...
- ...until problem discovered or new feature needed. Then, start over.



*based on Fig. 11.1b, p. 257 of course text.

Software Development Cycle

- Note: backward edges from every stage.
 - Later stages discover that earlier ones missed something.
 - Errors in design manifest as obstacles to implementation.
 - Customer feedback points out necessary changes.
- So-called **spiral** model.



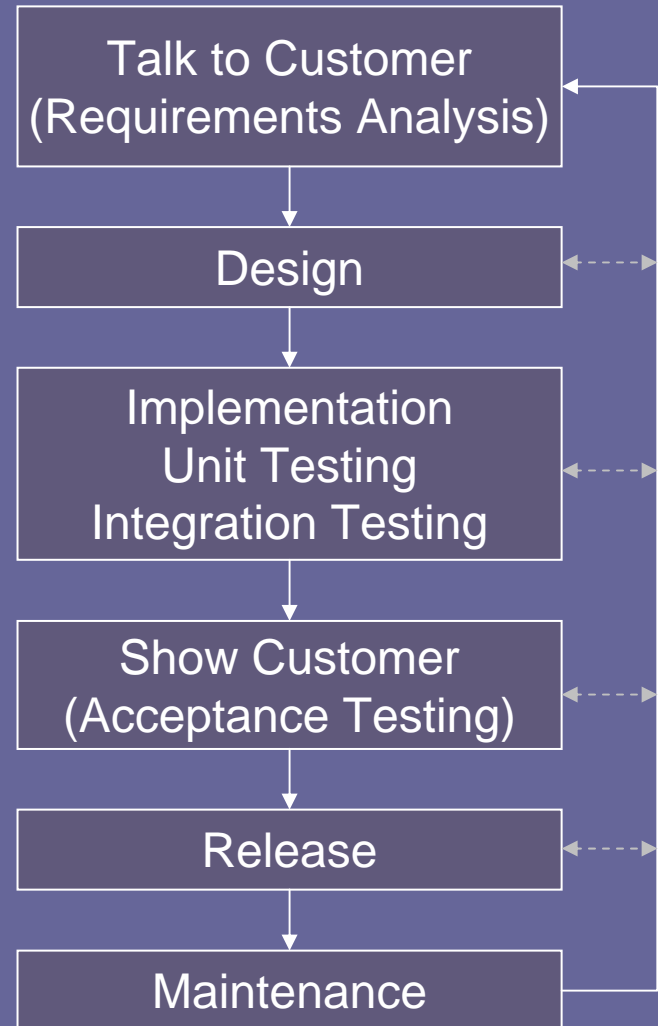
Waterfall vs. Iteration

Two main styles of process.

- **Waterfall.**

Try to do stages one at a time, in order, for whole system.

- Gather **all** requirements.
- Design **entire** system.
- Implement **entire** system.
- Acceptance-test **entire** system.



Waterfall vs. Iteration

Two main styles of process.

- **Waterfall.**

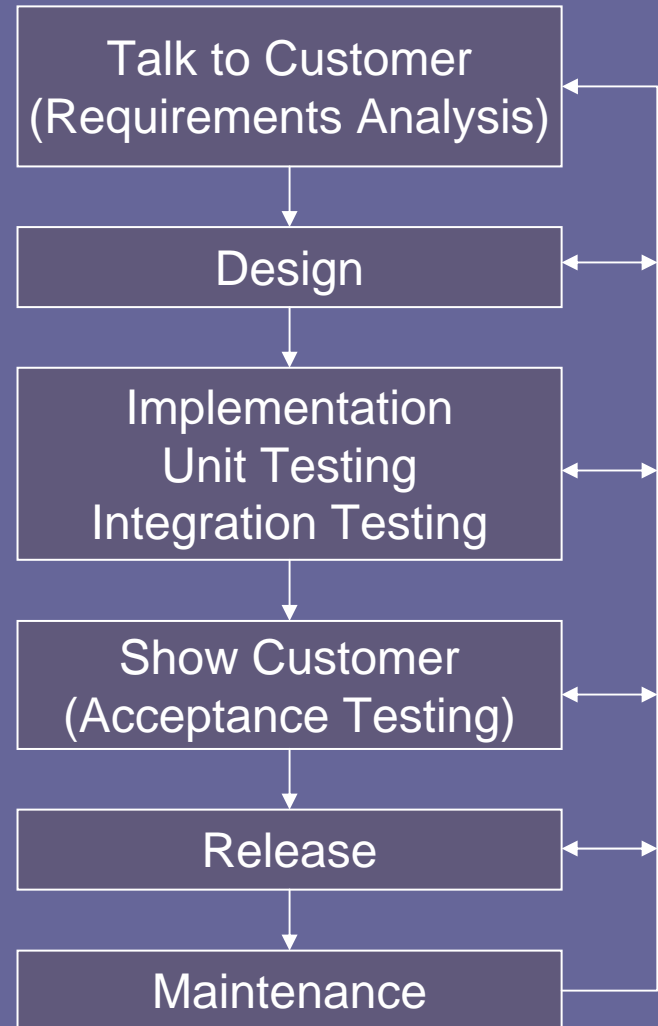
Try to do stages one at a time, in order, for whole system.

- Gather **all** requirements.
- Design **entire** system.
- Implement **entire** system.
- Acceptance-test **entire** system.

- **Iterative.**

Do whole cycle many times, adding features each time.

- Design/implement/test **small prototype**.
- Design/implement/test **additional features**.
- Repeat until system has all required features.



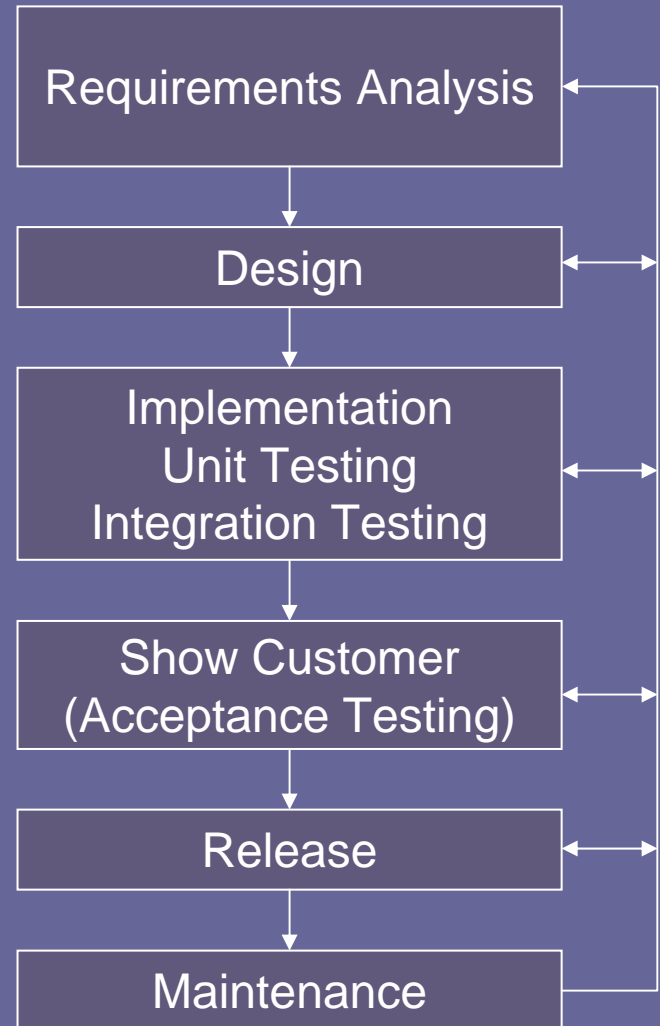
Waterfall vs. Iteration

- “Everyone knows” iterative process is better.
 - More frequent customer feedback opportunities.
 - Problems found sooner.
 - Each iteration must produce working (incomplete) system – easier to know if project is on schedule.
- So everyone claims to be doing iterative development even if they’re not.*

Requirements Analysis

Requirements

- First stage of any project.
“What should it do?”
- More specifically:
 - How it will be used.
 - How it should respond to errors.
 - User errors.
 - Software errors.
 - Hardware problems.
 - Performance requirements.
 - Changes anticipated in future versions.



Online Reading: Question

- Link on course homepage:
“Ask Slashdot” discussion (3-7-07):
“Getting Accurate Specifications for Software?”

“I design internal software for **users that are largely computer-illiterate**, and obtaining accurate specs for these programs has become a huge challenge. In the most recent instance, I asked for detailed specs on what an accounting program should do ... and received a **Word document mock-up of an input screen** ... When I asked my boss ... he responded saying that it was my responsibility to determine what was needed. How do I convey to the users that, **in order to develop the software they want, I need detailed, accurate specs?**”

Online Reading: Responses

- “count your lucky stars that your company is incapable of writing proper specs - if they were, **they would have outsourced your job** to India or Brazil a long time ago.”
- “You could try a kind of passive-aggressive approach. Keep misunderstanding them. A bit **like a monty python sketch**. Don't go so far that they really get angry. Judge it so they come to their senses and start to write down exactly what they want. ... **I think you are screwed**. Sorry.”
- “An analyst's job **is** to understand the business rules and figure out how they can be sanely implemented in an IT solution ... [T]he best **analysts are, first and foremost, generalists**: capable of quickly absorbing the rudiments of any computable field of human endeavor.”

Online Reading: Responses

- “It's **too big an intellectual challenge** for most people to think about the details of software design.”
- “I try to get them to tell me how they would do it with **a pencil and paper**. They won't answer the question as asked, of course. They'll say "I need some transaction where I can put..." or "there needs to be some file where..." - at this point you interrupt and ask them, again, how they would do it *with pencil and paper*. Eventually, you'll get to the answer. Then you, the developer/analyst, should be able to work out how to do it.”
- “[C]onsider your job more **like an architect**... If you think about what architects do for their clients, they figure out roughly what the client wants ... They then produce a number of different designs and design ideas to run past the client. Iterate a few times and then once they have sign off, build it.”

Online Reading: Responses

- “Basically, you camp out in the users "Den" and observe them, taking notes and trying to **understand how they work**, what buttons they push, which user interfaces frustrate them, which things they like, etc. You then take this back and use it to publish your requirements specs.”
- “Take the word doc and **bash together a prototype...**; drop the prototype in front of the users and make notes as they say "nooo! not like that, it needs to do X, Y and Z"; feed back into the prototype and try again. Finally use this prototype as a "living" requirements document. The hard part is persuading the pointy haired types that that prototype is, in fact, not the completed piece of software. Yeah, good luck with that.”

What was the point of that?

- Just for fun, really.
Don't take these people's advice too seriously.
- But points out a few things:
 - Engineers create software for non-engineers to use.
 - Engineers and non-engineers don't always communicate well.
 - The better you are at this, the more valuable you are to the customer.

Gather Background Information

- Your system may just be a new way to do something the customer already does.
 - Find out how they do it now.
 - Other systems yours must interact with.
 - Influence of customer's organizational structure, policies, regulations.
 - What should stay the same?
What should change?

Scenarios

- A **scenario** is a typical interaction between a user (or **actor**) and the system.
- Example (Google):
 - User opens `www.google.com` in browser, enters query string into text field and clicks Search button. Server responds with first page of results. User clicks on page numbers or Previous/Next to navigate between pages of results, then clicks a result link to go to the page it describes.
- Sequence of actions directed toward a particular **goal** (“Find a Web Site”).
- Other user actions correspond to different scenarios.
 - E.g., “I’m Feeling Lucky” or clicking on a sponsored link.

Scenarios

The program starts with a blank display component. User opens a file containing a Graph by choosing a particular menu item and selecting the file using a standard dialog box. This causes the graph to be displayed. The user chooses a starting and an ending vertex by selecting them from lists. The program immediately highlights the selected vertices and a minimum-weight path between them.

Many decisions recorded here could have been made differently.

Scenarios

The program starts with a blank display component. User opens a file containing a Graph by choosing a particular menu item and selecting the file using a standard dialog box. This causes the graph to be displayed. The user chooses a starting and an ending vertex by selecting them from lists. The program immediately highlights the selected vertices and a minimum-weight path between them.

(The program does not have a built-in default graph.)

Scenarios

The program starts with a blank display component. **User opens a file containing a Graph** by choosing a particular menu item and selecting the file using a standard dialog box. This causes the graph to be displayed. The user chooses a starting and an ending vertex by selecting them from lists. The program immediately highlights the selected vertices and a minimum-weight path between them.

(Rather than having to draw/enter the graph data.)

Scenarios

The program starts with a blank display component. User opens a file containing a Graph by choosing a particular menu item and **selecting the file using a standard dialog box**. This causes the graph to be displayed. The user chooses a starting and an ending vertex by selecting them from lists. The program immediately highlights the selected vertices and a minimum-weight path between them.

(Rather than having to type its name or choose from a predefined set of graphs.)

Scenarios

The program starts with a blank display component. User opens a file containing a Graph by choosing a particular menu item and selecting the file using a standard dialog box. This **causes the graph to be displayed**. The user chooses a starting and an ending vertex by selecting them from lists. The program immediately highlights the selected vertices and a minimum-weight path between them.

(Raises user-interface issues, e.g. what if graph is very large? Self-loops? Multiple edges?)

Scenarios

The program starts with a blank display component. User opens a file containing a Graph by choosing a particular menu item and selecting the file using a standard dialog box. This causes the graph to be displayed. The user chooses a starting and an ending vertex by **selecting them from lists**. The program immediately highlights the selected vertices and a minimum-weight path between them.

(Rather than typing their names, or clicking on them.)

Scenarios

The program starts with a blank display component. User opens a file containing a Graph by choosing a particular menu item and selecting the file using a standard dialog box. This causes the graph to be displayed. The user chooses a starting and an ending vertex by selecting them from lists. The program **immediately** highlights the selected vertices and a minimum-weight path between them.

(No need for user to click a “Go” button.)

Scenarios

The program starts with a blank display component. User opens a file containing a Graph by choosing a particular menu item and selecting the file using a standard dialog box. This causes the graph to be displayed. The user chooses a starting and an ending vertex by selecting them from lists. The program immediately highlights the selected vertices and a minimum-weight path between them.

(Rather than printing out the path as a list of vertices or edges.)

User Errors

- In scenarios just described, everything worked well.
- Also need to work out scenarios for user errors and/or exceptional situations.

User Errors

- In scenarios just described, everything worked well.
- Also need to work out scenarios for user errors and/or exceptional situations.
- Example:
 - Same as Google example before, except the server cannot find any pages matching the query. Server displays an error page offering some searching hints to the user. User can retry either by using browser's Back button or by typing a new query in search field on error page.

User Errors

- In scenarios just described, everything worked well.
- Also need to work out scenarios for user errors and/or exceptional situations.
- Example:
 - Same as Google example before, except the server cannot find any pages matching the query. Server displays an error page offering some searching hints to the user. User can retry either by using browser's Back button or by typing a new query in search field on error page.
- Examples:
 - Same as Graph example before, except the user cancels the file selection dialog, or manages to type a nonexistent file name, or selects a file that's not a Graph.
 - Also, suppose user selects start & end in wrong order, or tries to select vertices when no graph is open,...

Performance Requirements

- What kind of hardware will run the system?
 - Massive grid? Server? Notebook? Cell phone? Smart card?
 - Will **memory** be limited?
Is it a slow CPU? Fast? Parallel?
- Is **real-time** performance important?
 - Hard** real-time: avionics, weather forecasting, chess playing,...
 - Soft** real-time: user interaction, multimedia,...
- Is **availability** critical?
 - Web services, sales databases...
- Is it critical that data survive hardware failures?
- These things are always desirable, but not always worth the same amount of time/money.

Scale and Performance

The program starts with a blank display component. User opens a file containing a Graph by choosing a particular menu item and selecting the file using a standard dialog box. This causes the graph to be displayed. The user chooses a starting and an ending vertex by selecting them from lists. The program immediately highlights the selected vertices and a minimum-weight path between them.

What might be different if it were important to work well with very large graphs?

Scale and Performance

The program starts with a blank display component. User opens a file containing a Graph by choosing a particular menu item and selecting the file using a standard dialog box. This causes the graph to be displayed. The user chooses a starting and an ending vertex by selecting them from lists. The program immediately highlights the selected vertices and a minimum-weight path between them.

What might be different if it were important to work well with very large graphs?

Anticipate Changes

- Think about what might change in future versions.
- Helps in deciding where “abstraction boundaries” belong.

Anticipate Changes

- Think about what might change in future versions.
- Helps in deciding where “abstraction boundaries” belong.
- Example (Graph Viewer):
 - Support different file formats?
 - Graphs with different vertex/edge annotations?
 - Support for very large graphs?
 - Requiring more efficient data structures?
 - Too big to read entire graph into memory?
 - Too big to store entire graph on user’s disk?
 - Improved visual rendition? Draw node names? Edge weights?
 - Options to save/load paths?
 - Allow user to edit graph?

Aside: Predictability

- Should you *really* try to anticipate changes in requirements?
- Predictive vs. Adaptive planning.
 - Predictive: make a plan, follow it.
Know what you will deliver, how long it will take, what it will cost.
 - Adaptive: expect things to change.
- Some development processes discourage planning for change.
 - E.g.* Extreme Programming

Stock Tracker

- Liskov gives detailed example of analysis for a program to track investments.
- Won't describe it here, but you should read it.