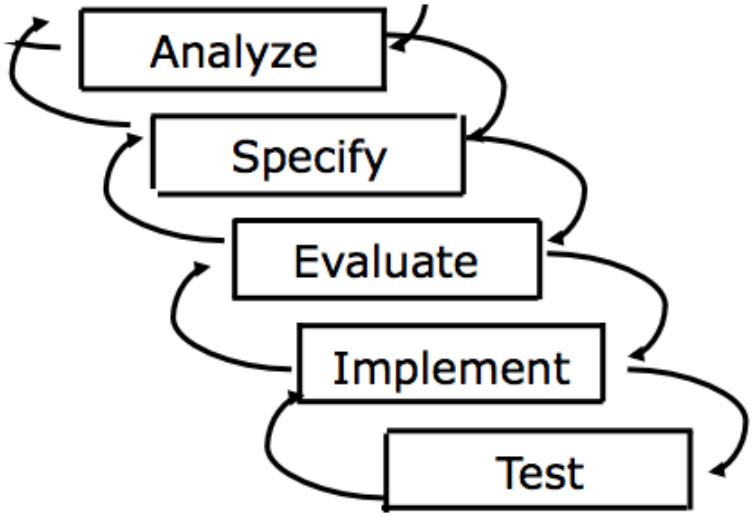
Session #6: Iterative and Waterfall User Experience Design

You need to understand waterfall

- Usability.gov: A linear design process whereby, steps are completed and the project passes onto the next phase with evaluation completed at the end.

Waterfall

- Waterfall is more linear/task based



What is Iterative?

- For UI & UX Design purposes, iterative does not mean “repetition”
- Mathematical Definition: “a problem is solved by proposing a series of candidate solutions, each one building on the preceding attempt, until a desired degree of accuracy is achieved.”

Deeper…

- Sometimes an attempt will take us in the wrong direction and we have to back up to a previous iteration and re-think the design.
- By executing this model rapidly, using rough online prototypes, and avoiding the trap of getting buried in content detail too soon, such mistakes are easy to correct.

Results…

- In evaluating prototype designs that were done relatively quickly, no one has yet invested too much time or resources to interfere with objectivity, multiple perspectives can create a collective wisdom of what will work best, and the team can continuously monitor project expectations of development time, budget, and desired outcomes to agree on what will be “good enough.”

Learn by doing…

- Every app is different
- *Until you have actually built what you are designing, you are not going to be able to fully understand it.*

Example of iterative design

The Danger of Iterative Design

Getting Reliable User Feedback from both users and clients

- The client may misinterpret our prototypes
- Cognitive Overload

How is it different from Agile?

- Agile is far more iterative
- Used for bigger projects, bigger teams
- Agile has a tighter feedback loop
- Client is more involved in Agile Design
- Agile has more testing, sooner.
- Benchmarks are larger with iterative design

So you might think agile is better.

- Agile costs money, but can save money
- Agile requires a change in work habits.
- Iterative design gets the project done more quickly, with more risk. For small teams who are experienced with working together it is a more appropriate solution.
- The faster the designs are produced, the more $$$$

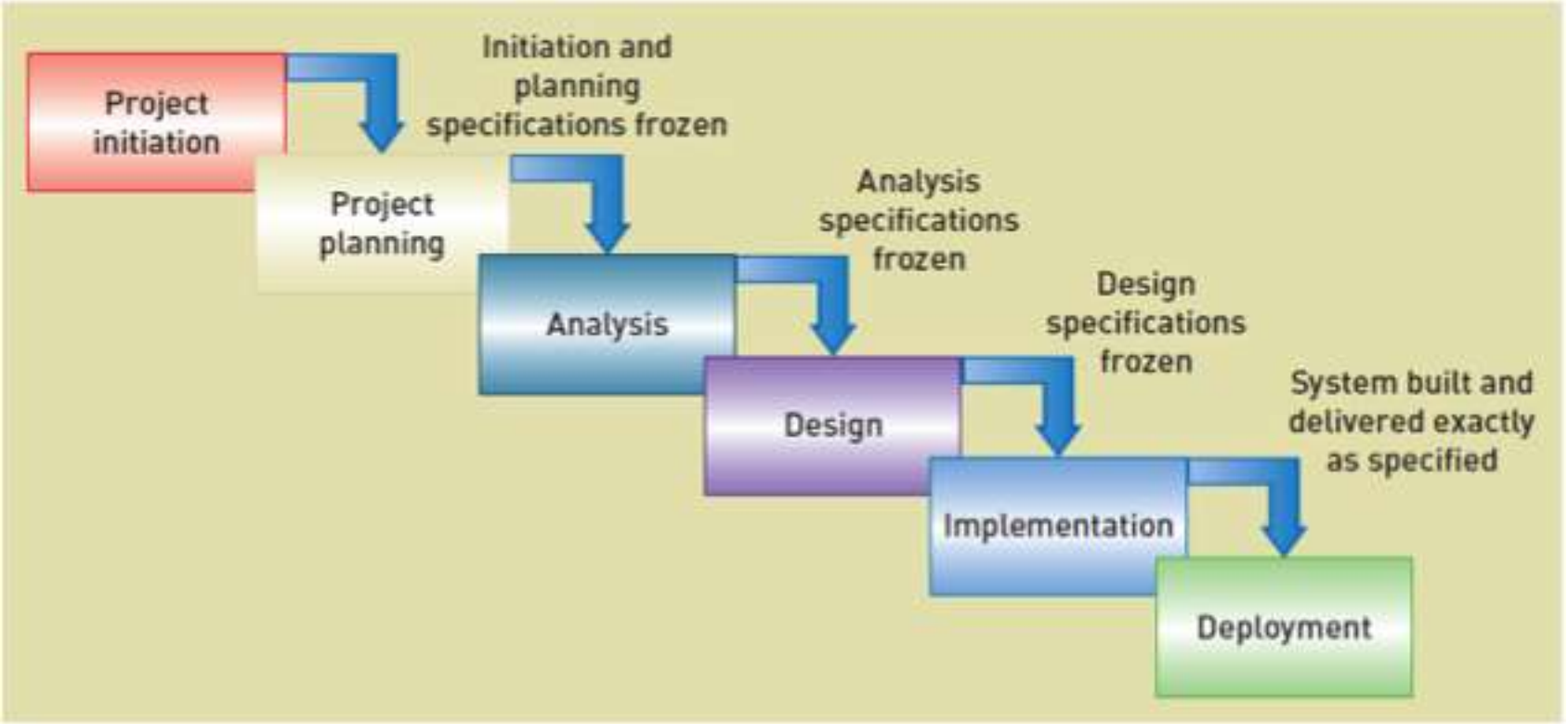
My case is rare…

- The fact that I use Iterative design is rare. The way I work would not work for most people. Sometimes, it even backfires on me, and then I have to recode or redesign something.

Less time documenting…

- Iterative design spends less time documenting, and more time producing.
- For smaller teams increasingly impossible to keep all the documents up to date, which means they can’t be relied on, which means all the work you put into them may be for naught.

Waterfall


Project initiation
Initiation and planning specifications frozen
Project planning
Analysis specifications frozen
Analysis
Design specifications frozen
Design
System built and delivered exactly as specified
Implementation
Deployment

Waterfall Approach

- Each life cycle phase is completed in **sequence** and then the results of the phase flow on to the next phase
- There is **no going back** once the phase is completed (like a waterfall) or it is extremely difficult to do
- The key **deliverables** for each phase are typically produced **on paper** (hundreds of pages in length)
- The decisions made at each phase are **frozen**, i.e. they cannot be changed

Waterfall evolution

- The biggest improvement of the waterfall model over previous (chaotic) approaches to software development is the discipline it puts on developers to think first, and code second.
- Requirements and designs generally precede the first line of code.

Pros and Cons

The two key advantages of the waterfall model:

- Identifying system requirements long before programming begins
- It minimizes changes to the requirements as the project proceeds

The key disadvantages:

- The design must be completely specified on paper before programming begins
- A long time elapses between the completion of the system proposal in the analysis phase and the delivery of the system (usually many months or years).
- A paper document is often a poor communication mechanism, so important requirements can be overlooked in the hundreds of pages of documentation
- Users rarely are prepared for their introduction to the new system, which occurs long after the initial idea for the system was introduced.
- If the project team misses important requirements, expensive post-implementation programming may be needed.
- A system may require significant rework because of changes in business environment since the time the analysis phase occurred. It means going back to the initial phases and following the changes through each of the subsequent phases in turn.

Validation is a weakness of waterfall:

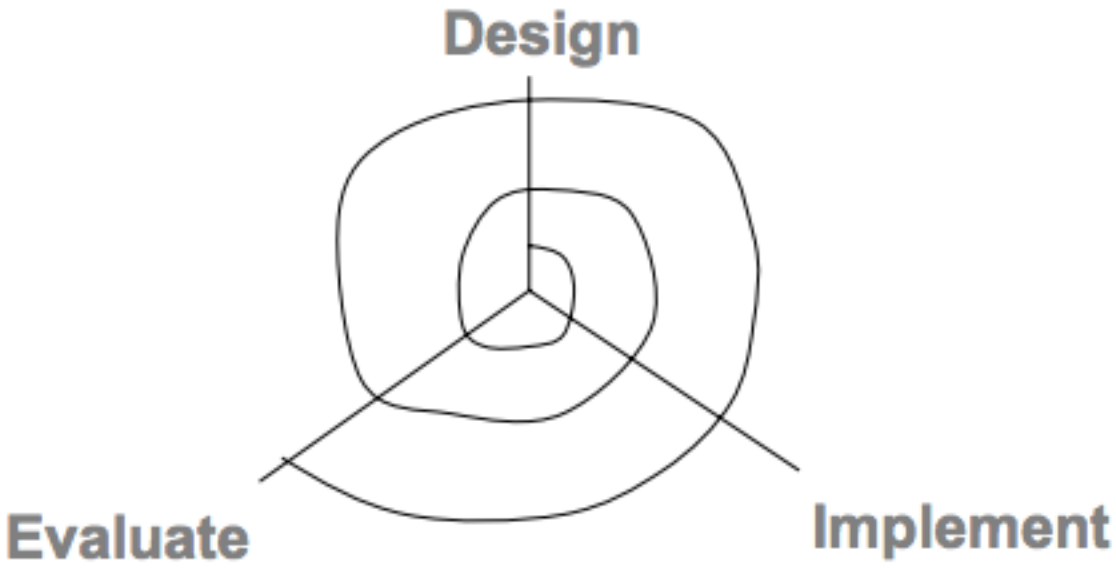
- Validation is not always sufficient; sometimes problems are missed until the next stage.

- Trying to code the design may reveal flaws in the design – e.g., that it can't be implemented in a way that meets the performance requirements. Trying to integrate may reveal bugs in the code that weren't exposed by unit tests. So the waterfall model implicitly needs feedback between stages.

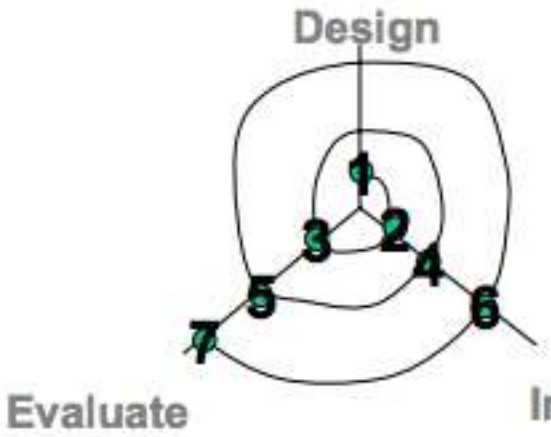
More dangers…

- The danger arises when a mistake in an early stage – such as a missing requirement – isn’t discovered until a very late stage – like acceptance testing. Mistakes like this can force costly rework of the intervening stages.

Spiral Model



Typical Project



1. Task analysis
2. Paper prototype
3. In-class user testing
4. Computer prototype
5. Heuristic evaluation
6. Implementation
7. User testing

Iterative with Spiral

- Early iterations use cheap prototypes
- – Parallel design is feasible: build & test multiple prototypes to explore design alternatives
- Later iterations use richer implementations, after UI risk has been mitigated
- Every prototype is evaluated
- – Users involved in all iterations
- More iterations generally means better UI
- Only mature iterations are seen by the world

Criteria for Agile/Iteration/Waterfall

- Short time schedules Projects with short time schedules are well suited for models as far as they are designed to increase the speed of development. Prototyping and phased development are excellent choices because they best enable the project team to adjust the functionality in the system. If the project schedule starts to slip, it can be readjusted by removing functionality from the version or prototype under development.
- Caution: The waterfall model does not allow for easy schedule changes.
- Schedule visibility One of the greatest challenges in systems development is knowing whether a project is on schedule. This is particularly true of the structured methods because design and implementation occur at the end of the project. The RAD models move many of the critical design decisions to an earlier point in the project to help project managers to recognize and address risk factors and keep expectations in check

Negative perceptions…

- Read this
- Many knock waterfall and say there is no place for this in UI Design.
- But let's not forget there is room for feedback loops

Many firms are evolving to agile.

- The biggest challenge is getting an employee buy in.
- Embracing the term “sprint”
- An important selling point is career growth
- Some do both agile and waterfall
- http://limina-ao.com/approach/waterfall.html & http://limina-ao.com/approach/agile.html

So what do we choose?

- Understand the pros and cons of these methodologies, can a blend be applied that satisfies your particular situation?
- Maybe run from a simple iterative model to an agile system when you have assets/personel

Case Study

- Case Study #2 due at the end of the week.
- Then we are back in production mode
- Wireframe due at then end of week 7
- Comp Due week 8
- Critiques in week 9
- Final Projects due week 10