Introduction to the Unified Process

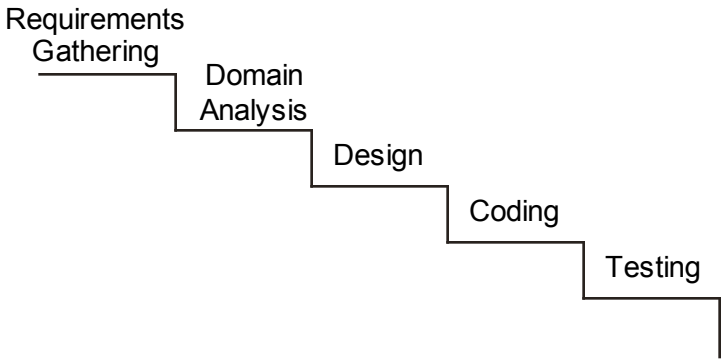
- The Traditional Waterfall Process
- The Benefits of Iteration
- Using UML in the Unified Process

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Tradition: The Waterfall

- The traditional process for software development involves several phases, each which must completed before the next phase can begin
- Potential problems include:
  - Analysis Paralysis
  - End Game Failure
  - Changed Requirements



2 - 2

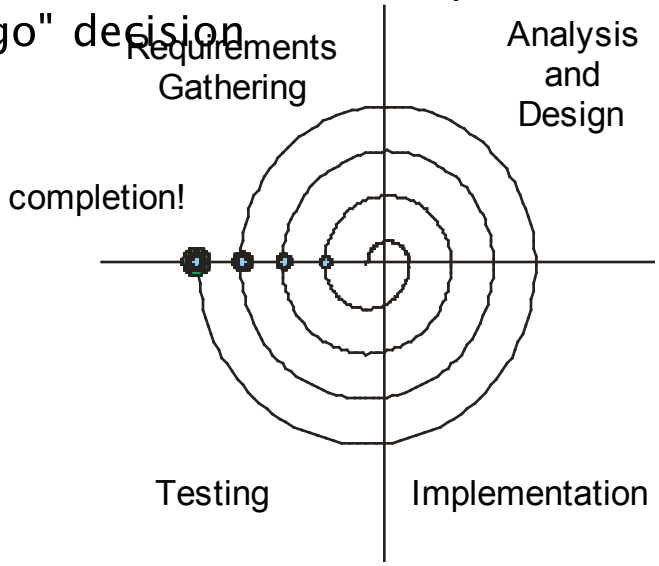
The sequential waterfall lifecycle model requires that each phase be fully completed before the next can begin. The potential problem is that if there are mistakes of omission or commission in an early phase, you don't discover them until the end. For example, if an end-user miscommunicates a feature request during requirements gathering, you won't know there's a problem until final testing. And that's an issue since it generally costs more to fix problems the further you go in the sequence.

This is not to say that the waterfall model is always inappropriate. In scenarios where absolute software quality is required and you have sufficient development time and budget, this model works well (an example is spacecraft software). However, for most business applications, the waterfall model often results in projects that are not finished on time or within budget.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

The Unified Process - Iterations

- The Unified Process emphasizes the notion of iterations, where each iteration creates a functional, non-feature-complete software system
- At the end of each iteration, you make a "go/no-go" decision



2 - 3

Instead of doing each phase sequentially, within an iterative process, you do some requirements gathering, some analysis and design, some coding and testing and end up with a working executable system that you can demonstrate to the stakeholders.

Each iteration adds functionality to the previous -- as you complete iterations, the system becomes more feature complete.

Note that during each iteration, you must perform testing -- both software testing and end-user acceptance testing. During each subsequent iteration, you must continue to perform regression testing to ensure that the additional function added in the iteration doesn't break existing work. Because of this, the Unified Process emphasizes automated "unit tests" that make it easy to protect against regressions.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Benefits of Iteration

- Early alleviation of high risk elements
- Early success or failure
- Early stakeholder feedback
- Each iteration improves the process
- Avoid analysis paralysis

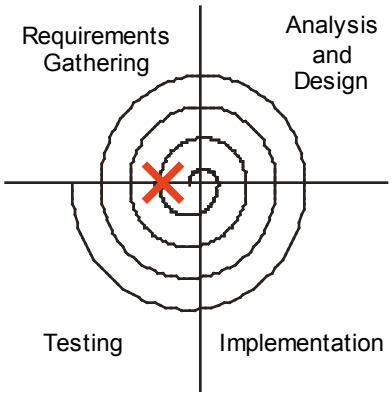
Unlike in the waterfall, with an iterative process you get relatively immediate results that you can demonstrate to the stakeholders, thus getting early feedback as to whether you are on the right track or not.

And it's important to note that iterative processes tend to improve as your development team becomes more adept at the process and learns what to do and what not to do.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Identifying Risk

- It's critical to perform the highest risk work in the first iteration(s)
- What defines high risk?
  - Core functionality
  - Least likely elements to successfully produce (hardest code)



2 - 5

To get the most benefit from an iterative process, it's crucial that you do the riskiest work first and that you make a go/no-go decision at the end of each iteration. If you do this properly, the chances of a complete failure are minimized, and if you do fail, you do so before you've spent a lot of time and budget on the project.

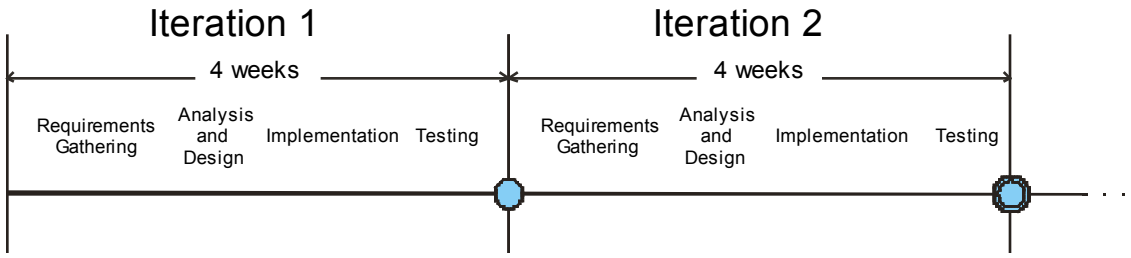
But how do you determine what's the riskiest part? There's no hard and fast rules, but generally, you look for the components that others depend on or for components that you are not sure if you have the technical expertise to complete.

The beauty of this approach is that you get a morale boost if the iteration is successful and gain confidence that the project can actually be completed.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Timeboxing the Iterations

- It's critical to set a hard limit for the duration of iterations -- omit features if time is running short
- Remember that each iteration produces a working executable
- The iteration duration should be about two to six weeks



2 - 6

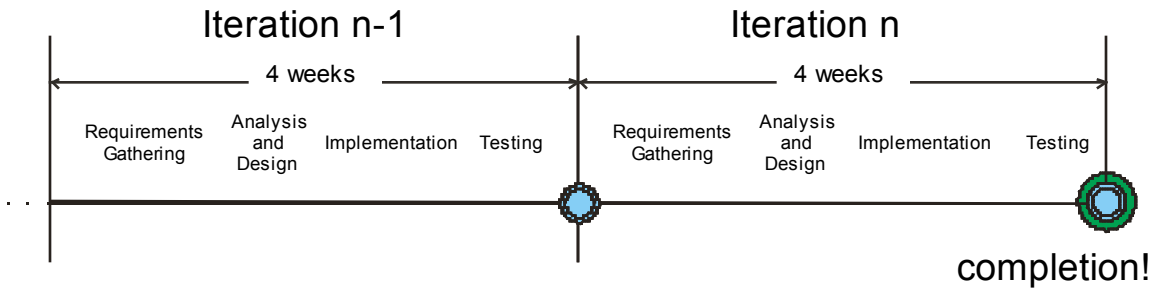
A potential danger of iterations is that if you extend the time for an iteration, you don't have the opportunity for the end-iteration acceptance testing, which is one of the main benefits of iteration. In other words, the longer you extend an iteration, you get closer to the waterfall model.

Thus, you should decide on the iteration length in advance and then stick to it, postponing features to the next iteration if you find time running out. The iteration length itself depends on a number of factors, especially the size of the team working on the project. The bigger the team, the longer you will need to get something done during an iteration, since bigger teams have more "friction" resulting from more communication and coordination of the team.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

How Do You Know When You are Done?

- You're finished when all of the use cases have been successfully implemented and tested
- Stakeholder (end-user) acceptance is crucial



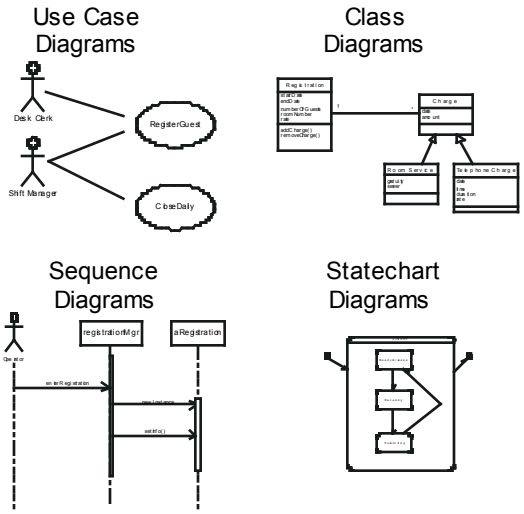
When the final iteration is complete, your software product successfully implements all of the uses of the system required by the stakeholders (end-users).

Of course, that doesn't mean the end of the program's lifecycle -- there's still maintenance and updates. However, if you've used good object-oriented analysis and design techniques, the cost of maintaining the system should be reduced.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Using UML in the Unified Process

- UML provides diagrams for each of the phases of an iteration
- Remember that each iteration involves requirements gathering, analysis, design, implementation and testing



The UML provides the notation and vocabulary for implementing a process like the Unified Process.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

The Amount of Ceremony

- Some development teams adopt a high-ceremony process where detailed diagrams and documents are required
- Other development teams adopt a more agile process that emphasizes the software product over the UML artifacts
- Your development team needs to decide on the amount of ceremony appropriate for your project

---

For more information on agile processes, go to http://www.agilealliance.com/home.

A high-ceremony process may be required by contract (e.g. government contract) or if you are producing software as a third party.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

You Know You Didn't Understand UP When

- You try to define most of the requirements before starting design or implementation
- You believe that a suitable iteration length is four months long
- You try to plan a project in detail from start to finish; you try to speculatively predict all of the iterations and what should happen in each one

---

These bullet points are taken from "Applying UML and Patterns" by Craig Larman, published by Prentice Hall.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Review Questions

- Discuss how the traditional waterfall can lead to analysis paralysis.
- Why is the go/no-go decision at the end of each iteration crucial?
- Discuss the factors that influence the iteration length.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Chapter Summary

In this chapter, you learned:

- About the benefits of an iterative process
- About timeboxing and identifying risk when planning iterations

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum