Object-oriented Analysis and Design

Stages in a Software Project

• Requirements Writing
  Understanding the Client’s environment and needs.

• Analysis
  Identifying the concepts (classes) in the problem domain and their static associations

• Design
  Identifying “solution domain” classes and mapping the functionality into these classes

• Implementation
  Constructing and individually testing the classes that compose the system.

• System Integration and Testing
  Testing the system as a whole and doing corrective maintenance

• Maintenance
  Adapting the system to a changing environment to extend its useful life
The Waterfall Lifecycle Approach

Thoroughly clarify, record, or implement each phase of the project before beginning the next. The subsequent phase is based upon the design documentation previously developed.
The Waterfall Lifecycle Approach

Problems with the Waterfall Approach

Large steps are taken in which many decisions are made without the benefit of feedback.

Requirements and Design decisions, once established, are frozen in place.

Speculative decisions increase and compound.

High risk or difficult problems are tackled late.

There is low adaptability for incorporating either design or implementation concepts “learned” in the development process into the project.
The Unified Process

The Unified Process is a software development process or methodology that above all promotes Iterative Development.

The result of each iteration is an executable, but incomplete system. The system may need many iterations before it is ready for production.

**Benefits of iterative development include:**

- Early mitigation of high risks
- Early visible progress.
- Early feedback, user engagement, and adaptation, leading to a system that more nearly meets the needs of the various stakeholders.
- Managed complexity – no compounding of complexity by postponing the implementation phase.
- Learning within an iteration.
Management of a Software Project

There are two groups involved with the management of a software project:

“Management” – whose concern is resource allocation, delivery dates, profit margin, etc.

Management likes the “benchmarks” inherent in the Waterfall approach. Due dates can be set, and resources allocated to each phase of the project, and project management documents can be completed according to a schedule.

Technical staff – whose concern is producing a well-engineered product within the constraints of the project.

The iterative process as emphasized in UP is a better approach for engineering a software product, but it less suited for producing project reports that indicate the status of the project and the completion of well-defined phases of the work important to management.
Timeboxing

Management of a UP project.

Iterations are “timeboxed” or fixed in length.

Iteration lengths of between two to six weeks are recommended.

Each iteration period has its own development plan.

If all the planned activities cannot be completed during an iteration cycle, the completion date should not be extended, but rather tasks or requirements from the iteration should be removed and added to the next iteration cycle.
Advantages of Timeboxing

Timeboxing provides four clear advantages over using more distant completion dates for major phases of the project.

Parkinson’s Law -- “Work expands to fill the time available for its completion”  Distant or fuzzy completion dates exacerbate this effect. If the end date for the next cycle is only two weeks away, it forces the team to focus and make important decisions immediately.

Prioritization and Decisiveness – Short timeboxed iterations force a team to make decisions regarding the priority of the work and the risks involved. If the immediate deadline is only short weeks away, there is no time to be vague. Concrete decisions about what will be accomplished within the iteration cycle have to be made.

Team Satisfaction – Short, timeboxed iterations lead to a quick and repeating sense of accomplishment and closure.

Stakeholder confidence – When the team makes a commitment to producing something tangible within a short period of time, stakeholders develop a greater sense of confidence in the team and satisfaction with the company.
The Unified Process

The UP organizes work and iterations across four major phases

Inception – approximate vision, business case, scope, vague estimates.

NOT a requirements phase, but a feasibility phase

Elaboration – refined vision, iterative implementation of the core architecture, resolution of high risks, identification of most requirements and scope, more realistic estimates.

A phase where the core architecture is iteratively designed and implemented

Construction – iterative implementation of the remaining lower risk and easier elements, preparation for deployment.

Transition – beta tests and deployment
The Unified Process

The end of each iteration is a minor release.
The Unified Process

UP Disciplines

The UP groups related work activities into Disciplines. Disciplines are a set of work activities (and related artifacts) within one subject area such as requirements analysis.

A single discipline such as requirements analysis occurs in varying degrees across many iterations, and an iteration will incorporate many different Disciplines.
What About UML?

UML is a standard diagramming notation. It is NOT object-oriented analysis and design, but just a standard for visualizing and documenting the artifacts of software systems that is used during OOA/OOD.

UML tools will be used within the various disciplines during each iteration.
Alternative Methodologies

The UP is an “agile process”. It allows the development team to adapt to changes in requirements or technology that occur during the development process.

Other Examples of an Agile Approach include:

• Boehm’s Spiral Development Process
• Incremental Development
• Extreme Programming

The product is developed and delivered incrementally.
Boehm’s Spiral Model

Determine objectives, alternatives, and constraints

Evaluate alternatives, identify, resolve risks

Review

Requirements plan
Life-cycle plan

Risk analysis

Development plan

Integration and test plan

Prototype 1

Prototype 2

Prototype 3

Concept of operation

S/W requirements

Simulations, models, benchmarks

Product design

Detailed design

Design V & V

Unit test

Integration test

Acceptance test

Code

Service

Operational Prototype

Develop, verify next level product

Plan next phase
What’s the Difference?

How do these various agile processes differ?

The difference is mainly one of emphasis.

Extreme Programming (EP) emphasizes “test-first programming”. Write a unit test *before* writing the code to be tested.

- Write a small test
- Write a piece of the code
- Make it pass the test
- Repeat until unit is is complete.

Extreme Programming also emphasizes continuous integration. New code is integrated into the entire system as soon as it is “checked-in”.
A representation of the difference between UP and EP

The Unified Process  Extreme Programming

Specification  Build, test, and integrate the
Design  units into the system
Code
Test

Iterations
Formal Methods

The software requirements specification is refined into a detailed formal specification which is represented in a mathematical notation.

The development processes – design, implementation, unit testing – are replaced by a transformational development process where the formal specification is refined, through a series of transformations, into a program.

Formal methods require specialized expertise, and do not scale up beyond specialized domains very readily.