The Unified Process

- Until recently, three of the most successful object-oriented methodologies were:
  - Rumbaugh’s OMT
  - Booch’s method (Rational Rose)
  - Jacobson’s Objectory

The Unified Process (2)

- In 1999, Booch, Rumbaugh, and Jacobson published a complete object-oriented analysis and design methodology that unified their three separate methodologies.
  - Original name: Rational Unified Process (RUP)
  - Next name: Unified Software Development Process (USDP)
  - Name used today: Unified Process (for brevity)

The Unified Process (3)

- The Unified Process is not a series of steps for constructing a software product.
  - No such single “one size fits all” methodology could exist.
  - There is a wide variety of different types of software.
- The Unified Process is an adaptable methodology.
  - It has to be modified for the specific software product to be developed.

The Unified Process (4)

- UML is graphical:
  - A picture is worth a thousand words.
- UML diagrams enable software engineers to communicate quickly and accurately.

Iteration and Incrementation within the Object-Oriented Paradigm

- The Unified Process is a modeling technique:
  - A model is a set of UML diagrams that represent various aspects of the software product we want to develop.
- UML stands for unified modeling language:
  - UML is the tool that we use to represent (model) the target software product.
Iteration and Incrementation within the Object-Oriented Paradigm (2)
- The object-oriented paradigm is iterative and incremental in nature
  - There is no alternative to repeated iteration and incrementation until the UML diagrams are satisfactory

Iteration and Incrementation within the Object-Oriented Paradigm (3)
- The version of the Unified Process in this book is for
  - Software products small enough to be developed by a team of three students during the semester or quarter
- However, the modifications to the Unified Process for developing a large software product are also discussed

Iteration and Incrementation within the Object-Oriented Paradigm (4)
- The goals of this book include:
  - A thorough understanding of how to develop smaller software products
  - An appreciation of the issues that need to be addressed when larger software products are constructed
  - We cannot learn the complete Unified Process in one semester or quarter
    - Extensive study and unending practice are needed
    - The Unified Process has too many features
    - A case study of a large-scale software product is huge

Iteration and Incrementation within the Object-Oriented Paradigm (5)
- The book covers much, but not all, of the Unified Process
  - The topics covered are adequate for smaller products
- To work on larger software products, experience is needed
  - This must be followed by training in the more complex aspects of the Unified Process

Requirements Workflow
- The aim of the requirements workflow
  - To determine the client’s needs
- First, gain an understanding of the application domain (or domain, for short)
  - That is, the specific business environment in which the software product is to operate
- Second, build a business model
  - Use UML to describe the client’s business processes
  - If at any time the client does not feel that the cost is justified, development terminates immediately

Requirements Workflow (2)
- It is vital to determine the client’s constraints
  - Deadline
    - Software products are often mission critical
  - Parallel running
  - Portability
  - Reliability
  - Rapid response time
  - Cost
    - The client will rarely inform the developer how much money is available
    - A bidding procedure is used instead
Requirements Workflow (3)
- The aim of this concept exploration is to determine
  - What the client needs
  - Not what the client wants

Analysis Workflow
- The aim of the analysis workflow
  - To analyze and refine the requirements
- Why not do this during the requirements workflow?
  - The requirements artifacts must be totally comprehensible by the client
  - The artifacts of the requirements workflow must therefore be expressed in a natural (human) language
  - All natural languages are imprecise

Analysis Workflow (2)
- Example from a manufacturing information system:
  - "A part record and a plant record are read from the database. If it contains the letter A directly followed by the letter Q, then calculate the cost of transporting that part to that plant"
- To what does it refer?
  - The part record?
  - The plant record?
  - Or the database?

Analysis Workflow (3)
- Two separate workflows are needed
  - The requirements artifacts must be expressed in the language of the client
  - The analysis artifacts must be precise, and complete enough for the designers

The Specification Document
- Specification document ("specifications")
  - It constitutes a contract
  - It must not have imprecise phrases like "optimal," or "98% complete"
- Having complete and correct specifications is essential for
  - Testing and
  - Maintenance

The Specification Document (2)
- The specification document must not have
  - Contradictions
  - Omissions
  - Incompleteness
- In the Unified Process, the "document" is the UML diagrams and descriptions
Software Project Management Plan
- Once the client has signed off the specifications, detailed planning and estimating begins
- We draw up the software project management plan, including
  - Cost estimate
  - Duration estimate
  - Deliverables
  - Milestones
  - Budget
- This is the earliest possible time for the SPMP

The Design Workflow
- The aim of the design workflow is to refine the analysis workflow until the material is in a form that can be implemented by the programmers
  - Many nonfunctional requirements need to be finalized at this time, including
    - Choice of programming language
    - Reuse issues
    - Portability issues

Classical Design
- Architectural design
  - Decompose the product into modules
- Detailed design
  - Design each module:
    - Data structures
    - Algorithms

Object-Oriented Design
- Classes are extracted during the object-oriented analysis workflow and
  - Designed during the design workflow
- Accordingly
  - Classical architectural design corresponds to part of the object-oriented analysis workflow
  - Classical detailed design corresponds to part of the object-oriented design workflow

The Design Workflow (2)
- Retain design decisions
  - For when a dead-end is reached
  - To prevent the maintenance team reinventing the wheel

The Implementation Workflow
- The aim of the implementation workflow is to implement the target software product in the selected implementation language
  - A large software product is partitioned into subsystems
  - The subsystems consist of components or code artifacts
The Test Workflow

- Testing is carried out in parallel with all other workflows
- The test workflow is the responsibility of:
  - Every developer and maintainer, and
  - The quality assurance group
- Traceability of artifacts is an important requirement for successful testing

Requirements Artifacts

- Every item in the analysis artifacts must be traceable to an item in the requirements artifacts
  - Similarly for the design and implementation artifacts

Analysis Artifacts

- The analysis artifacts should be checked by means of a review
  - Representatives of the client and analysis team must be present
- The SPMP must be similarly checked
  - Pay special attention to the cost and duration estimates

Design Artifacts

- Design reviews are essential
  - A client representative is not usually present
- Look for faults
  - Logic faults
  - Interface faults
  - Lack of error processing
  - Nonconformance to specifications

Implementation Artifacts

- Each component is tested as soon as it has been implemented
  - Unit testing
- At the end of each iteration, the completed components are combined and tested
  - Integration testing
- When the product appears to be complete, it is tested as a whole
  - Product testing
- Once the completed product has been installed on the client’s computer, the client tests it
  - Acceptance testing

Implementation Artifacts (2)

- COTS software is released for testing by prospective clients
  - Alpha release
  - Beta release
- There are advantages and disadvantages to being an alpha or beta release site
Postdelivery Maintenance

- Postdelivery maintenance is an essential component of software development
  - More money is spent on postdelivery maintenance than on all other activities combined
- Problems can be caused by
  - Lack of documentation of all kinds

Postdelivery Maintenance (2)

- Two types of testing are needed
  - Testing the changes made during postdelivery maintenance
    - Regression testing
- All previous test cases (and their expected outcomes) need to be retained

Retirement

- Software can be unmaintainable because
  - A drastic change in design has occurred
  - Interdependencies have built up
  - The product must be implemented on a totally new hardware/operating system
  - Documentation is missing or inaccurate
  - Thus, it may be cheaper to rewrite the software from scratch than to modify it
- These are instances of maintenance (rewriting of existing software)

Retirement (2)

- True retirement is a rare event
- It occurs when the client organization no longer needs the functionality provided by the product