The Software Process

Xiaojun Qi

The Unified Process

• Until recently, three of the most successful object-oriented methodologies were
  – Booch’s method
  – Jacobson’s Objectory
  – Rumbaugh’s OMT (Object Modeling Technique)

• Today, the unified process is usually the primary choice for object-oriented software production. That is, the unified process is the primary object-oriented methodology.

The Unified Process (Cont.)

• In 1999, Booch, Jacobson, and Rumbaugh published a complete object-oriented analysis and design methodology, which is called Unified Process. It 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 (Cont.)

• The Unified Process is not a specific series of steps for constructing a software product since
  – There is a wide variety of different types of software products
  – No such single “one size fits all” methodology could exist

• The Unified Process is an adaptable methodology
  – It has to be modified for the specific software product to be developed

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 to be developed

• UML stands for unified modeling language
  – UML is the tool that we use to represent (model) the target software product
  – UML is graphical since a picture is worth a thousand words

• UML diagrams enable software engineers to communicate more quickly and accurately than if only verbal descriptions were used.

Iteration and Incrementation Within the Object-Oriented Paradigm (Cont.)

• The object-oriented paradigm is iterative and incremental in nature
  – Each workflow consists of a number of steps, and to carry out that workflow, the steps of the workflow are repeatedly performed until the members of the development team are satisfied that they have an accurate UML model of the software product they want to develop.
  – There is no alternative to repeated iteration and incrementation until the UML diagrams are satisfactory
Iteration and Incrementation Within the Object-Oriented Paradigm (Cont.)

• The goals of this book include:
  – A thorough understanding of how to develop smaller software products by a team of three students during the semester
  – 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, since
  – 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

Core Workflows of the Unified Process: The Requirements Workflow

• The aim of the requirements workflow is to determine the client’s needs
  – Gain an understanding of the application domain, that is, the specific business environment in which the target software product is to operate
  – 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

The Requirements Workflow (Cont.)

• The preliminary investigation of the client’s needs is called concept exploration

• It is vital to determine exactly what the client needs and to find out the client’s constraints
  – Deadline
  – Reliability
  – Cost
    • The client will rarely inform the developer how much money is available
    • A bidding procedure is used instead
  – Parallel running, Portability, and Rapid response time

Core Workflows of the Unified Process: The Analysis Workflow

• The aim of the analysis workflow is to analyze and refine the requirements to achieve the detailed understanding of the requirements essential for developing a software product correctly and maintaining it easily.

• 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

The Analysis Workflow (Cont.)

• All natural languages are imprecise! An 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?

• 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 Analysis Workflow: Specification Document

• Specification document (“specifications”)
  – Constitutes a contract
  – It must not have imprecise phrases like “optimal,” or “98 percent complete”

• Having complete and correct specifications is essential for both testing and maintenance

• The specification document must not have
  – Ambiguity
  – Contradictions
  – Incompleteness
The Analysis Workflow: Software Project Management Plan

• Once the client has signed off the specifications, detailed planning and estimating begins

• The software project management plan includes:
  – Deliverables: what the client is going to get
  – Milestones (Duration estimate): when the client gets them
  – Budget (Cost estimate): how much it is going to cost

• This is the earliest possible time for the SPMP

Core Workflows of the Unified Process: 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

• Retain design decisions
  – To backtrack and redesign certain pieces when a dead-end is reached
  – To prevent the maintenance team reinventing the wheel

Classical vs. Object-Oriented Design

• Classical Design
  – Architectural design
    • Decompose the product into modules
  – Detailed design for each module
    • Data structures
    • Algorithms

• Object-Oriented Design
  – Classes are extracted during the object-oriented analysis workflow, and designed during the design workflow
  – Classical architectural design corresponds to part of the object-oriented analysis workflow
  – Classical detailed design corresponds to part of the object-oriented design workflow

Core Workflows of the Unified Process: 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 smaller subsystems, which are implemented in parallel by coding teams
  – The subsystems consist of components or code artifacts implemented by an individual programmer

The Test Workflow -- Traceability

• 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
The Test Workflow
-- Traceability

- **Design Artifacts:**
  - Reviews are essential
    - A client representative is not usually present
- **Implementation Artifacts:**
  - Unit testing: Each component is tested as soon as it has been implemented
  - Integration testing: At the end of each iteration, the completed components are combined and tested
  - Product testing: When the product appears to be complete, it is tested as a whole
  - Acceptance testing: Once the completed product has been installed on the client’s computer, the client tests it
  - COTS software is released for testing by prospective clients: Alpha release and 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
- Two types of testing are needed
  - Testing the changes made during postdelivery maintenance
  - Regression testing: Make sure that the functionality of the rest of the product has not been compromised.
- 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
  - Many changes may have been made to the original design
    - Interdependencies inadvertently have been built into the product.
    - Documentation is missing or inaccurate
  - The product must be implemented on a totally new hardware/operating system.
    - It may be cheaper to rewrite the software
- True retirement is a rare event; It occurs when the client organization no longer needs the functionality provided by the product

The Phases of the Unified Process

- The four increments are labeled as:
  - Inception phase
  - Elaboration phase
  - Construction phase
  - Transition phase
- The phases of the Unified Process correspond to increments
- In theory, there could be any number of increments
  - In practice, development seems to consist of four increments
- Every step performed in the Unified Process falls into
  - One of the five core workflows and also
  - One of the four phases

The Phases of the Unified Process (Cont.)

- Why does each step have to be considered twice?
  - Workflow
    - Technical context of a step
  - Phase
    - Business (i.e., economic) context of a step
- Example: One step to determine the client’s needs is to build a business model.
  - Building a business model should be presented within both technical and economic contexts.
The Inception Phase

- The aim of the inception phase is to determine whether the proposed software product is economically viable. There are several steps for this phase:
  1. Gain an understanding of the domain
  2. Build the business model
     - Understand precisely how the client organization operates in the domain
  3. Delimit the scope of the proposed project
     - Focus on the subset of the business model that is covered by the proposed software product
  4. Begin to make the initial business case

The Inception Phase: Requirements Workflow -- The Initial Business Case (Sample Questions)

- Is the proposed software product cost effective?
  - Have the necessary marketing studies been performed?
  - How long will it take to obtain a return on investment?
  - What will be the cost not to develop the product?
- Can the proposed software product be delivered in time?
  - What will be the impact if the product is delivered late?
- What are the risks involved in developing the software product, and how can these risks be mitigated?
  - Does the team have the necessary experience?
  - Is new hardware needed for this software product?
    - If so, is there a risk that it will not be delivered in time?
    - If so, is there a way to mitigate that risk, perhaps by ordering back-up hardware from another supplier?
  - Are software tools (Ch. 5) needed? Are they currently available? Do they have all the necessary functionality?

The Inception Phase: Requirements Workflow -- Risks

- There are three major risk categories:
  - Technical risks
    - See earlier slide
  - Not getting the requirements right
    - Mitigated by performing the requirements workflow correctly
  - Not getting the architecture right
    - The architecture may not be sufficiently robust
- To mitigate all three classes of risks
  - The risks need to be ranked so that the critical risks are mitigated first

The Inception Phase: Analysis, Design, Implementation, Testing Workflows

- A small amount of the analysis workflow may be performed
  - Information needed for the design of the architecture is extracted
- A small amount of the design workflow may be performed
- Coding is generally not performed. However, a proof-of-concept prototype is sometimes built to test the feasibility of constructing part of the software product
- The test workflow commences almost at the start of the inception phase
  - The aim is to ensure that the requirements have been accurately determined

The Inception Phase: Planning

- There is insufficient information at the beginning of the inception phase to plan the entire development
  - The only planning that is done at the start of the project is the planning for the inception phase itself
- Similarly, the only planning that can be done at the end of the inception phase is the plan for the elaboration phase

The Inception Phase: Documentation

- The deliverables of the inception phase include:
  - The initial version of the domain model
  - The initial version of the business model
  - The initial version of the requirements artifacts
  - A preliminary version of the analysis artifacts
  - A preliminary version of the architecture
  - The initial list of risks
  - The initial ordering of the use cases (Chapter 10)
  - The plan for the elaboration phase
  - The initial version of the business case
The Inception Phase:
The Initial Business Case

• Obtaining the initial version of the business case is the overall aim of the inception phase

• This initial version incorporates
  – A description of the scope of the software product
  – Financial details
  – For marketed product, the business case will include
    • Revenue projections, market estimates, initial cost estimates
  – For in-house product, the business case will include
    • The initial cost–benefit analysis

Elaboration Phase

• The aim of the elaboration phase is to
  – Refine the initial requirements
  – Refine the architecture
  – Monitor the risks and refine their priorities
  – Refine the business case
  – Produce the project management plan

• The major activities of the elaboration phase are refinements or elaborations of the previous phase

The Tasks of the Elaboration Phase

• The tasks of the elaboration phase correspond to:
  – All but completing the requirements workflow
  – Performing virtually the entire analysis workflow
  – Starting the design of the architecture

The Elaboration Phase: Documentation

• The deliverables of the elaboration phase include:
  – The completed domain model
  – The completed business model
  – The completed requirements artifacts
  – The completed analysis artifacts
  – An updated version of the architecture
  – An updated list of risks
  – The project management plan (for the rest of the project)
  – The completed business case

Construction Phase

• The aim of the construction phase is to produce the first operational-quality version of the software product
  – This is sometimes called the beta release

• The emphasis in this phase is on
  – Implementation, and
  – Testing
    • Unit testing of modules
    • Integration testing of subsystems
    • Product testing of the overall system

The Construction Phase: Documentation

• The deliverables of the construction phase include:
  – The initial user manual and other manuals, as appropriate
  – All the artifacts (beta release versions)
  – The completed architecture
  – The updated risk list
  – The project management plan (for the remainder of the project)
  – If necessary, the updated business case
The Transition Phase

- The aim of the transition phase is to ensure that the client's requirements have indeed been met
  - Faults in the software product are corrected
  - All the manuals are completed
  - Attempts are made to discover any previously unidentified risks

- This phase is driven by feedback from the site(s) at which the beta release has been installed

The Transition Phase: Documentation

- The deliverables of the transition phase include:
  - All the artifacts (final versions)
  - The completed manuals

One- and Two-Dimensional Life-Cycle Models

Why a Two-Dimensional Model?

- A traditional life cycle is a 1D model
  - Represented by the single axis
  - Example: Waterfall model

- The Unified Process is a 2D model
  - Represented by the two axes
  - Example: Evolution Tree Model

- The two-dimensional figure shows
  - The workflows (technical contexts), and
  - The phases (business contexts)

Why a Two-Dimensional Model? (Cont.)

- Are all the additional complications of the 2D model necessary?
  - In an ideal world, each workflow would be completed before the next workflow is started
  - In reality, the development task is too big for this
  - As a consequence of Miller’s Law
    - The development task has to be divided into increments (phases)
    - Within each increment, iteration is performed until the task is complete

Why a Two-Dimensional Model? (Cont.)

- At the beginning of the process, there is not enough information about the software product to carry out the requirements workflow and other core workflows

- A software product has to be broken into subsystems

- Even subsystems can be too large at times
  - Components may be all that can be handled until a fuller understanding of all the parts of the product as a whole has been obtained
Why a Two-Dimensional Model? (Cont.)

• The Unified Process handles the inevitable changes well
  – The moving target problem
  – The inevitable mistakes

• The Unified Process is the best solution to date for treating a large problem as a set of smaller, largely independent subproblems
  – It provides a framework for incrementation and iteration
  – In the future, it will inevitably be superseded by some better methodology

Improving the Software Process

• The fundamental problem with software
  – The software process is badly managed

• Software process improvement initiatives
  – Capability maturity model (CMM)
  – ISO 9000-series
  – ISO/IEC 15504

Capability Maturity Models (CMM)

• CMM is not a life-cycle model. It is a set of strategies for improving the software process
  – SW–CMM for software
  – P–CMM for human resources ("people")
  – SE–CMM for systems engineering
  – IPD–CMM for integrated product development
  – SA–for software acquisition

• These strategies are unified into CMMI (capability maturity model integration)

SW–CMM

• A strategy for improving the software process

• Put forward in 1986 by the SEI (Software Engineering Institute)

• Fundamental ideas:
  – Improving the software process (activities, techniques, and tools) leads to
    • Improved software quality
    • Delivery on time, within budget
  – Improved management leads to
    • Improved techniques

SW–CMM (Cont.)

• Five levels of maturity are defined
  – Maturity is a measure of the goodness of the process itself

• An organization advances stepwise from level to level

• It takes:
  – 3 to 5 years to get from level 1 to level 2
  – 1.5 to 3 years from level 2 to level 3
  – SEI questionnaires highlight shortcomings and suggest ways to improve the process

Level 1: Initial Level

• Ad hoc approach
  – The entire process is unpredictable
  – Management consists of responses to crises

• Most organizations world-wide are at level 1
Level 2: Repeatable Level

- Basic software management
  - Management decisions should be made on the basis of previous experience with similar products
  - Measurements ("metrics") are made
    - These can be used for making cost and duration predictions in the next project
  - Problems are identified, immediate corrective action is taken

Level 3: Defined Level

- The software process is fully documented
  - Managerial and technical aspects are clearly defined
  - Continual efforts are made to improve quality and productivity
  - Reviews are performed to improve software quality
  - CASE tools are applicable now (and not at levels 1 or 2)

Level 4: Managed Level

- Quality and productivity goals are set for each project
  - Quality and productivity are continually monitored
  - Statistical quality controls are in place to enable management to distinguish a random deviation from a meaningful violation of quality or productivity standards.

Level 5: Optimizing Level

- Continuous process improvement
  - Statistical quality and process controls are used to guide the organization
  - Feedback of knowledge from each project to the next

Key Process Areas

- There are key process areas (KPAs) for each level
- Level-2 KPAs include:
  - Requirements management
  - Project planning
  - Project tracking
  - Configuration management
  - Quality assurance
- Compare
  - Level 2: Detection and correction of faults
  - Level 5: Prevention of faults

Other Software Process Improvement Initiatives: ISO 9000

- A set of five standards for industrial activities
  - ISO 9001 for quality systems
  - ISO 9000-3, guidelines to apply ISO 9001 to software
  - There is an overlap with CMM, but they are not identical
    - ISO: There is a stress on documenting the process to ensure consistency and comprehensibility
    - ISO Philosophy: Adherence to the standard does not guarantee a high-quality product but rather reduces the risk of a poor-quality product.
Other Software Process Improvement Initiatives: ISO/IEC 15504

- Original name: Software Process Improvement Capability Determination (SPICE)
  - Includes process improvement, software procurement
  - Extends and improves CMM, ISO 9000
  - A framework, not a method
    - CMM, ISO 9000 conform to this framework
    - Now referred to as ISO/IEC 15504
    - Or just 15504 for short

Costs and Benefits of Software Process Improvement

- Hughes Aircraft moved from level 2 to 3:
  - Savings: $2M per year
- Raytheon moved from level 1 to 3
  - Productivity doubled
  - Return of $7.70 per dollar invested
- Tata Consultancy Services (India) used ISO 9000 and CMM
  - Errors decreased from 50% to 15%
  - Effectiveness of reviews increased from 40% to 80%

Costs and Benefits of Software Process Improvement (Cont.)

- There is interplay between
  - Software engineering standards organizations, and
  - Software process improvement initiatives
- ISO/IEC 12207 (1995) is a full life-cycle software standard
- In 1998, IEEE/EIA 12207 was published that incorporated ideas from CMM
- ISO 9000-3 now incorporates part of ISO/IEC 12207