CS 315 - Lecture 3 - Aug 26, 2015

Chapter 3: The Software Process

- Lecture Slides
- The Unified Process
    - In 1999, Booch, Jacobson, and Rumbaugh 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 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
- Phases = Increments
- Iterations - each version of the software at different stages in its development
- Workflows - activities spread all through the software's life span
- Workflows
    - Summary
        - Requirements
            - Analysis of app domain
            - Creation of requirement artifacts
        - Design
            - Creation of Solution and Design Artifacts
        - Implementation
            - Creation of the code
        - Testing
            - Assessment of processes and products
        - Deployment

        - Transition of system to user

    - Environment
        - Maintenance (communication and configuration management)

- Requirements Workflow
    - The aim of the requirements workflow is to determine the client's ***needs***
    - Getting an understanding of the *application domain* (or *domain* for short).
    - Second build a business model.
        - Use UML to describe the business processes.
        - If at any time the client does not feel that the cost is justified, development terminates immediately.

    - Determine the client's constraints
        - Deadline
            - Usually in Months
            - 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

- Analysis Workflow
    - **Goal**: Analyze and refine the requirements
    - So why not do this in the Requirements workflow?
        - Requirements must be totally understandable by the client
        - They are therefore expressed in natural language, which is imprecise
        - Analysis artifacts must be precise and complete enough for designers

    - 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
        - Maintenance

- Design Workflow
    - **Goal**: Refine the analysis workflow until the material is in a form that can be implemented by the programmers
        - Specification: what the program has to do
        - Design: how it should do it
    - Architecture Design
        - Modules, communication, reliability, security, portability
    - Detail Design
        - Algorithms, data structures, programming language(s), re-use
    - Object Oriented (Analysis and) Design
        - The promise of object oriented design is that it can more closely model the real world problem space
        - Identify classes and their relationships
        - Keep record of design decisions
            - To backtrack if dead-end is reached
        - Design should be open-ended
            - Future enhancements should be possible and facilitate maintenance
- Implementation Workflow
    - **Goal**: Implement the target software product in the selected implementation language(s)
    - At this point, all design decisions have been made. All there is left to do is implement the system
    - Large software is partitioned into sub-systems
        - Components and Code Artifacts
        - Divide-and-Conquer
    - The implementation of each code artifact is assigned to a programer (or team). If artifact A relies on artifact B, then programmer A and programmer B should communicate and know about their dependencies
    - The integration of the individual artifacts is crucial
        - Validates the define
    - Multiple releases may be necessary
        - Alpha release
        - Beta Release
        - Release Candidates (Microsoft, for example)
- Testing Workflow

        - Testing is the responsibility of
            - Every developer and maintainer
            - The quality assurance team (QA)
        - All artifacts from all phases must be traceable
            - Every module, class, method must be traced back to a design artifact, which is tracked back to an analysis artifact, which is traced back to a requirement
            - Crucial for testing
        - Requirement: Every software artifact must be traceable back to the requirements
            - Client reviews requirements
        - Analysis
            - Reviewed jointly
            - Client's expert and analysis team
        - Design
            - Reviewed by developers and QA team
        - Implementation
            - Unit Testing: each implemented component must be tested as soon as complete
            - Integration testing: After each iteration, combine components and test
            - System testing: Test software as a whole
            - Acceptance Testing: by client after software is installed
    - Maintenance and Retirement
        - Maintenance
        - Typically after the first version of the software is deployed and installed
            - But maintenance issues should be thought of early on in design and implementation
            - Longest and most costly of all workflows
            - Problems typically caused by lack of documentation
            - When a modification is made to the software, all tests (or some of them) must be re-run
                - Regression Testing
        - Retirement
            - Final stage of the software life span
            - Usually after many years of service
            - Causes of retirement
                - Client does not need the functionality provided by the software
                - Drastic change in design needed
                - Software must be implemented on totally new hardware

- Phases
    - Phase Summary
        - Inception
            - Scope
            - Use Cases
        - Elaboration
            - Initial architecture design
            - Cost and Resource estimates
        - Construction
            - Build components
            - Release
            - Acceptance Criteria
        - Transition
            - Deployment
    - Inception Phase
        - **Goal**: Determine whether it is worthwhile to develop the proposed software
            - Gain understanding of the domain
            - Build business model
            - Delimit scope of project
            - Begin initial business case
        - Business Case: Questions that should be answered
        - Risk
            - Three major risk categories
                - Technical Risk
                    - Competency
                    - Hardware/software acquirement
                - The risk of not getting the architecture right
                    - The architecture may not be sufficiently robust for later additions
                - The risk of not getting the requirements right
                    - Performing the requirements workflow correctly
            - Rank risks by order of criticality (and likelihood of occurrence)
            - All questions should be answered by the end of the inception phase

    - Inception Tasks
        - Small amount of architecture design should be extracted
        - No coding is done at this point
            - Proof-of-concept prototypes can be useful to asses Feasibility of parts of the software
            - Testing should start on requirements

    - Inception Deliverables
        - Initial version of the domain model
        - Initial version of the business model
        - Initial version of the requirements artifacts
        - A preliminary version of the analysis artifacts
        - A preliminary version of the architecture
        - Initial list of risks
        - Initial ordering of the use cases
        - Plan for the elaboration phase
        - Initial version of the business case

- Elaboration Phase
    - **Goal**: Refine and elaborate what was done in the Inception phase

        - Refine Architecture
        - Monitor risks and refine their priorities
        - Refine business case
        - Produce software project management plan
        - Elaboration tasks
            - Complete the requirements workflow
            - Perform almost the entire analysis workflow
            - Start the design of the architecture
            - Set up the development and testing environments

        - Elaboration Deliverables
            - 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

    - Implementation Phase
        - **Goal**: produce the first operational-quality version of the software
        - Tasks
            - Emphasis is mainly on Implementation
            - Testing

        - Construction Deliverables
            - The initial user manual and other manuals
            - All the artifacts (beta release versions)
            - The completed architecture
            - The updated risk list
            - The revised project management plan
            - If necessary, the updated business case

    - Transition Phase
        - **Goal**: ensure that the client's requirements have indeed been met

            - Faults in the software product are corrected
            - All manuals are completed
            - Attempts are made to discover any previously unidentified risks

        - Driven by feedback from the beta release
        - Deliverables
            - All the artifacts in their final version

- One vs Two Dimensional Model
    - One dimension is not really an accurate model
    - Two Dimensional is more descriptive and prescriptive of how software development works
        - Much more accepting of multiple processes going on at the same time