Process Models/Development Methodologies & Rational Unified Process

Topic # 16

Chapter 2 – Craig Larman

Process

Defines Who is doing What, When to do it, and How to reach a certain goal.

- Workers, the 'who'
- Activities, the 'how'
- Artifacts, the 'what'
- Workflows, the 'when'

What is a Process Model ?

It is a description of

i) what tasks need to be performed in

ii) what sequence under

iii) what conditions by

iv) whom to

achieve the "desired results."

Software Processes & Process Model

- **Software Process** is comprehensible sets of activities for specifying, designing, implementing and testing software systems.
- ***A (software/system) process model*** *is a description of the sequence of activities carried out in an SE project, and the relative order of these activities*

Common Activities of Process Model

Many different software processes but all involve:

- Specification – defining what the system should do;
- Design and implementation – defining the organization of the system and implementing the system;
- Validation – checking that it does what the customer wants;
- Evolution – changing the system in response to changing customer needs.

Software Process Models

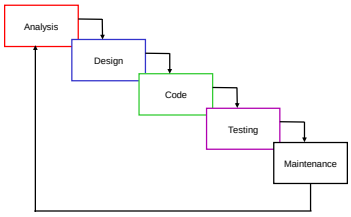
- ***The waterfall model***
  - *Separate and distinct phases of specification and development*
- ***Evolutionary development***
  - *Specification and development are interleaved*
- ***Incremental development***
  - *Major requirements must be defined; however, details can evolve with time.*
- ***Agile development***
  - *Requirements and solutions evolve through collaboration between self-organizing, cross-functional teams.*

Software Development Model

Waterfall

Linear sequence of stages/phases
Each box represents a **set of tasks** that results in the production of each or more work products.

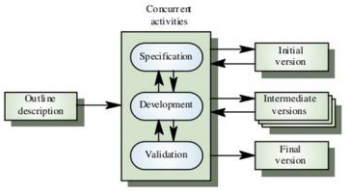
Each new phase begins when the **work products** of the previous phase as completed, frozen and signed off.



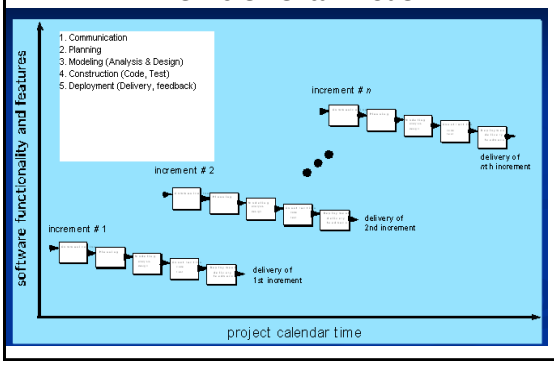
Evolutionary/Iterative development Model

- Software evolves over a period of time.
- Its an idea of considering initial requirements and developing initial implementation of the system also called as **initial version**.
- The implementation is then discussed with the customer and his comments are analyzed and **intermediate version** is built and released.
- The implementation is later fine tuned till the customer is happy with the system and the **final version** is released.

EVOLUTIONARY MODEL



The Incremental Model



The Incremental Model

- When initial requirements are reasonably well defined, but the overall scope of the development effort precludes a purely linear process. A compelling need to expand a limited set of new functions to a later system release.
- It combines elements of linear and parallel process flows. Each linear sequence produces deliverable increments of the software.
- The first increment is often a core product with many supplementary features. Users use it and evaluate it with more modifications to better meet the needs.

Agile Development Methodologies

- In English, Agile means 'ability to move quickly and easily' and responding swiftly to change – this is a key aspect of Agile software development as well.
- Agile software development is a group of software development methodologies where requirements and **solutions evolve** through collaboration between self-organizing, cross-functional teams.
- Agile methods break tasks into small increments with minimal planning and do not directly involve long-term planning.
- Iterations are short time frames (timeboxes) that typically last from one to four weeks.

Characteristics of Agile

Iterative and incremental

- Many short time boxed iterations with incremental releases. The iterative approach produces earlier release and better stakeholders' feedback.

Test driven development:

- The working software is the primary measure of progress in the project. Thus, each iteration release is tested to be working software.

People oriented

- Agile methodology believes quality of the people is most important in making sure the project is successful.

Lightweight:

- Agile methodology believes that the most efficient and effective communication is face to face conversation. Development team does not build large documents and control points.

Unified Process Model

- A process model that was created 1997 to give a framework for Object-oriented Software Engineering
- UML driven iterative process model (framework).
  - Maps out when and how to use different UML techniques
- OO Methodology for large scale software.
- Deliver value to customer.
- Accommodate change early on in project.
- Work as one team.
- Iterative, incremental model to adapt to specific project needs
- Risk driven development
- Combining spiral and evolutionary models

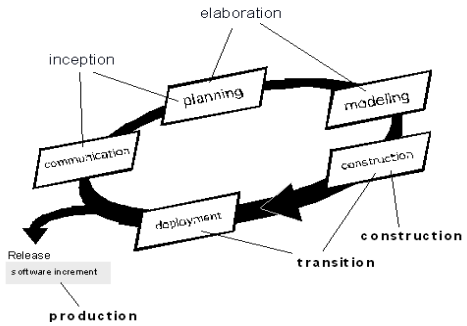
The Unified Process

- **2D process : phases and workflows**
- The Unified Process IS A 2-dimensional systems development process described by a
  - set of phases and (dimension one)
  - Workflows (dimension two)

The Unified Process

- Phases
  - Describe the business steps needed to develop, buy, and pay for software development.
  - The business increments are identified as phases
- Workflows
  - Describe the tasks or activities that a developer performs to evolve an information system over time.

Rational Unified Process Overview



The Unified Process

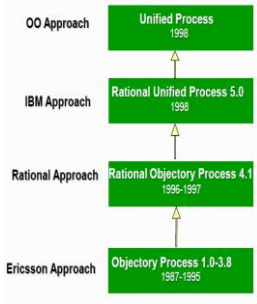
- **Utilizes Miller's Law**
  - Human concentration on 7 chunks of information.
  - To handle larger chunks, use stepwise refinement:
    - Concentrate on most important aspects.
    - Postpone aspects that are less critical
    - Results in product on small portions (incrimination)

Unified Process

- The Unified Process is not simply a process, but rather an extensible framework which should be customized for specific organizations or projects.
- The Rational Unified Process is, similarly, a customizable framework. As a result, it is often impossible to say whether a refinement of the process was derived from UP or from RUP, and so the names tend to be used interchangeably.

History of Creating the Unified Process

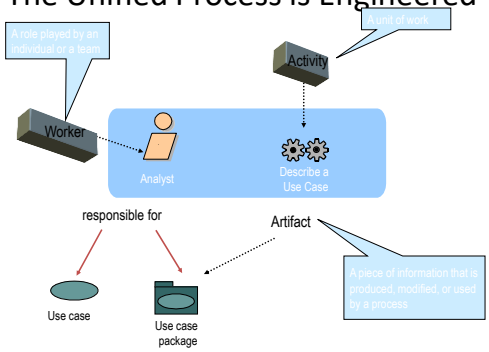
- Some roots in "Spiral Model " of Barry Boehm
- Core initial development around 1995-1998
- Canadian Air Traffic Control project as test bed
- Rational Corporation had commercial product in mind (RUP, now owned by IBM) but also reached out to public domain (UP)



Unified Process

- 1999 : Booch, Jacobson, and Rumbaugh combines 3 separate methodologies to form OOAD methodology – called UP.
- The Unified Process is an **adaptable methodology**
  - It has to be modified for the specific software product to be developed
- The Unified Process is a **modeling technique.** Model?
  - model is a set of UML diagrams that represent various aspects of software
  - UML is graphical. A picture is worth a thousand words
  - UML diagrams enables us to communicate quickly and accurately.
- The object-oriented paradigm is **iterative and incremental in nature.**

The Unified Process is Engineered

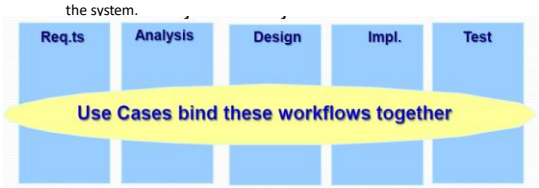


Basic Characteristics of Unified Process

- Object-oriented
- Use-case driven
- Architecture centric
- Risk focused
- Iteration and incrimination

UP Characteristics

- **Object-oriented**
  - Utilizes object oriented technologies.
  - Classes are extracted during OOA and designed during OOD.
- **Use-case driven**
  - Utilizes use case model to describe complete functionality of the system
  - This practice reinforces the fundamental notion that a system must conform to the needs of the users, instead of your users conforming to the system.



UP Characteristics

Architecture centric

- Focus core architecture in the early iterations
- In earliest iterations, get high valued requirements
- View of the whole design with the important characteristics made more visible
- Expressed with class diagram

Risk-focused:

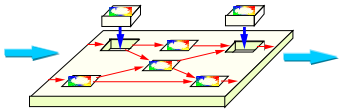
- The Unified Process requires the project team to focus on addressing the most critical risks early in the project life cycle. The deliverables of each iteration, especially in the Elaboration phase, must be selected in order to ensure that the greatest risks are addressed first.

UP Characteristics

Iterative and incremental

- Way to divide the work
- Iterations are steps in the process, and increments are growth of the product
- The basic software development process is iterative
  - Each successive version is intended to be closer to its target than its predecessor

The Unified Process is a Process Framework



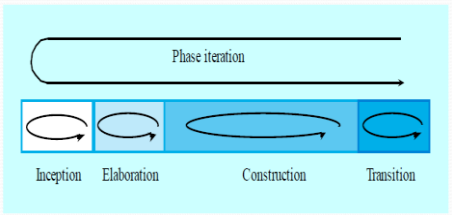
There is NO Universal Process!

- The Unified Process is designed for flexibility and extensibility
  - » allows a variety of lifecycle strategies
  - » selects what artifacts to produce
  - » defines activities and workers
  - » models concepts
  - » It is a process framework for developement

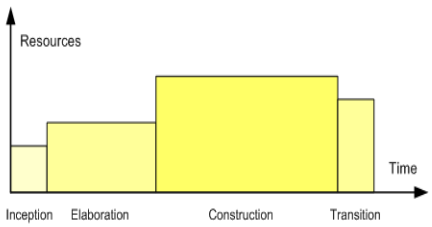
Rational Unified Process (RUP)

- RUP is a method of managing OO Software Development.
- It is a software development framework which is extensible and it features:
  - Iterative development
  - Requirement management
  - Component based architectural version.
  - Visual modeling of systems
  - Quality management
  - Change control management

Unified Process Model



Profile of a typical project showing the relative sizes of the four phases of the Unified Process.



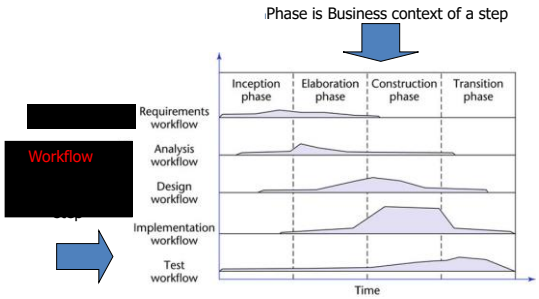
Unified Process Model

- Integrating two seemingly contradicting insights:
  - Definitive activities and deliverables as in the Waterfall Model.
  - Iterative and incremental processes.
  - Perform a mini waterfall project that ends with a a delivery of something tangible in code.
  - The result of a single iteration is an incremental improvement.
- A project is split into several *phases:*
  - Each phase is split into several *iterations.(2 to 6 weeks)*
  - Each iteration consists of the traditional process activities, known as *workflow.*
- Each workflow places *different* emphasis on the activities depending on the current iteration.
- Risk analysis is performed in each iteration – each iteration is risk driven.
- Slowly chip away at the project risks
  - Performance risks
  - Integration risks (different vendors, tools etc.)
  - Conceptual risks (hunt out design and analysis flaws)

Unified Process Phases

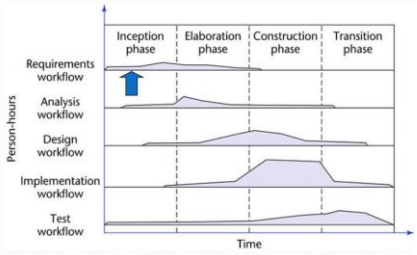
- Inception
  - Establish the business case for the system, define risks, obtain 10% of the requirements, estimate next phase effort.
- Elaboration
  - Develop an understanding of the problem domain and the system architecture, risk significant portions may be coded/tested, 80% major requirements identified.
- Construction
  - System design, programming and testing. Building the remaining system in short iterations.
- Transition
  - Deploy the system in its operating environment. Deliver releases for feedback and deployment.

The Phases/Workflows of the Unified Process



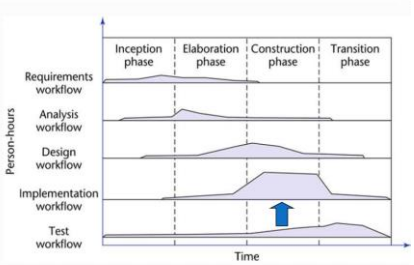
Phases/Workflows of the Unified Process

- **NOTE: Most of the requirements work or workflow is done in the inception phase.**
- **However some is done later.**



Phases/Workflows of the Unified Process

- **NOTE: Most of the implementation work or workflow is done in construction**
- **However some is done earlier and some later.**



Unified Process – Inception Activities

Inception

Project Initiation

- Start with an idea
- Specify the end-product vision
- Analyze the project to assess scope
- Work the business case for the project including overall costs and schedule, and known risks
- Identify Stakeholders
- Obtain funding

Unified Process – Inception Activities & Deliverables

Inception

- **Activities**
  - Project Planning
  - Build Team
  - Define initial iteration
  - Assess project risks and risk mitigation plan
- **Primary deliverables:**
  - A vision document
  - Any needed models or artifacts such as an use-case model (10%-20% complete),domain model, business model, or requirements and analysis artifacts.
  - Project plan, with phases and iterations with a more detailed plan for the elaboration phase.
  - A project glossary
  - One or several prototypes.

Unified Process – Elaboration Activities

Elaboration

- The goal of the elaboration phase is to baseline the most significant requirements.
- **Elaboration Essential Activities**
- Analyze the problem domain.
- Refine the Vision to understand the most critical Use Cases
- Create and baseline iteration plans for construction phase.
- Refine component architecture and decide build/buy/reuse
- Develop a project plan and schedule.
- Mitigate high-risk elements identified in the previous phase.

Unified Process – Elaboration Deliverables

Elaboration

- **Primary deliverables:**
  - Requirements model for the system
  - The completed domain model (use cases, classes)
  - The completed business model (costs, benefits,risks)
  - The completed requirements artifacts
  - The completed analysis artifacts
  - Updated Architectural model
  - Software project management plan

Unified Process – Construction Activities

Construction

- The goal of the construction phase is to clarify the remaining requirements and complete the development of the first operational quality version of the software product.
- Construction Essential Activities
  - Complete component development and testing (Integrate all remaining components and features into the product
  - Assure resource management control and process optimization

Unified Process – Construction Deliverables

Construction

- Primary deliverables
  - Working software system (beta release version)
  - Associated documentation
  - Acceptance testing documentation
  - User Manuals

Unified Process – Transition Objectives

Transition

- Deploy the system in its operating environment. Deliver releases for feedback and deployment.
- The focus of the Transition Phase is to ensure that software is available for its end users and meets their needs.
- Transition Objectives
  - Assess deployment baselines against acceptance criteria
- Primary deliverable
  - Final product onto a production platform
- Other deliverables
  - All the artifacts (final versions)
  - Completed manual

Phase Deliverables Summary

Inception Phase	Elaboration Phase	Construction Phase	Transition Phase
• 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 • The plan for the elaboration phase • The initial version of the business case	• 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	• 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	• All the artifacts (final versions) • The completed manuals

UP Life cycle in four phases

- Inception
- Elaboration
- Construction
- Transition
- The Enterprise Unified Process (EUP) adds two more phases to this:
  - Production: keep system useful/ productive after deployment to customer
  - Retirement: archive, remove or reuse etc.

Some examples role in UP

- Stake holders: customer, product manager etc.
- Software Architect: establish and maintains architectural vision
- Process Engineer: leads definition and refinement of development use case
- Graphic artist: assist in user interface design etc.

Some UP guidelines

- Attack risks early on and continuously so, before they will attack you
- Stay focused on developing executable software in early iterations
- Prefer component-oriented architectures and reuse existing components
- Quality is a way of life, not an afterthought

Six best "must" UP practices

1. Time-boxed iterations
2. *Strive for cohesive architecture* and reuse existing components: e.g.
   1. core architecture developed by small, co-located team
   2. then early team members divide into sub-project leaders
3. Continuously verify quality: *test early & often*, and realistically by integrating all software at each iteration

Six best "must" UP practices

4. Visual modeling: prior to programming, do at least some visual modeling to explore creative design ideas
5. Manage requirements: find, organize, and track requirements through skillful means
6. Manage change:
   - disciplined configuration management protocol, version control,
   - change request protocol
   - baselined releases at iteration ends

Unified Process Workflows

The Unified Process

- The Unified Process IS A
- 2-dimensional systems development process described by a
  - set of phases and (dimension one)
  - Workflows (dimension two)

The Unified Process

- Phases
  - Describe the business steps needed to develop, buy, and pay for software development.
  - The business increments are identified as phases
- Workflows
  - Describe the tasks or activities that a developer performs to evolve an information system over time.

Process Overview

	Phases (time)			
Workflow (tasks)	Inception	Elaboration	Construction	Transition
Requirements				
Analysis				
Design				
Implementation				
Test				

Static workflows in the Rational Unified Process

Workflow	Description
Business modelling	The business processes are modelled using business use cases.
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements.
Analysis and design	A design model is created and documented using architectural models, component models, object models and sequence models.
Implementation	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.

Static workflows in the Rational Unified Process

Workflow	Description
Testing	Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
Deployment	A product release is created, distributed to users and installed in their workplace.
Configuration and change management	This supporting workflow managed changes to the system
Project management	This supporting workflow manages the system
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

Primary workflows

- Unified process primary workflows:
  - Requirements workflow
  - Analysis workflow
  - Design workflow
  - Implementation workflow
  - Test workflow
  - Post delivery maintenance workflow
- Supplemental workflows:
  - Planning workflow

Planning workflow

- Define scope of Project
- Define scope of next iteration
- Identify Stakeholders
- Capture Stakeholders expectation
- Build team
- Assess Risks
- Plan work for the iteration
- Plan work for Project
- Develop Criteria for iteration/project closure/success
- UML concepts used: initial Business Model, using class diagram

Requirements workflow

- Primary focus
  - To determine the client's needs by eliciting both functional and nonfunctional requirements
- Gain an understanding of the *application domain*
- Described in the language of the customer

Workflow (tasks)
Requirements
Analysis
Design
Implementation
Testing

Requirements workflow

- The aim is to determine the client's needs
- First, gain an understanding of the *domain*
  - How does the specific business environment work
- 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
- It is vital to determine the client's constraints
  - Deadline -- Nowadays software products are often mission critical
  - Parallel running
  - Portability
  - Reliability
  - Rapid response time
  - Cost
- The aim of this *concept exploration* is to determine
  - What the client needs, and
  - *Not* what the client wants

Workflow (tasks)
Requirements
Analysis
Design
Implementation
Testing

Requirements workflow

- List candidate requirements
  - textual feature list
- Understand system context
  - domain model describing important concepts of the context
  - business modeling specifying what processes have to be supported by the system using Activity Diagram
- Capture functional and nonfunctional requirements
  - Use Case Model
- Supplementary requirements
  - physical, interface, design constraints, implementation constraints

Workflow (tasks)
Requirements
Analysis
Design
Implementation
Testing

Analysis workflow

- Primary focus
  - Analyzing and refining the requirements to achieve a detailed understanding of the requirements essential for developing a software product correctly
- To ensure that both the developer and user organizations understand the underlying problem and its domain
- Written in a more precise language

Workflow (tasks)
Requirements
Analysis
Design
Implementation
Testing

Analysis workflow

- The aim of the analysis workflow
  - To analyze and refine the requirements
- 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
- 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
  - Testing, and
  - Maintenance
- The specification document must not have
  - Contradictions
  - Omissions
  - Incompleteness

Workflow (tasks)
Requirements
Analysis
Design
Implementation
Testing

Analysis workflow

- Structure the Use Cases
- Start reasoning about the internal of the system
- Develop Analysis Model: Class Diagram and State Diagram
- Focus on what is the problem not how to solve it
- Understand the main concepts of the problem
- Three main types of classes stereotypes may be used:
  - Boundary Classes: used to model interaction between system and actors
  - Entity Classes: used to model information and associated behavior deirectly derived from real-world concept
  - Control Class: used to model business logic, computations transactions or coordination.
- The specification document must not have
  - Contradictions
  - Omissions
  - Incompleteness

Workflow (tasks)
Requirements
Analysis
Design
Implementation
Testing

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
  - Determines the internal structure of the software product

Workflow (tasks)
Requirements
Analysis
Design
Implementation
Testing

Design workflow

The goal 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 using data structures and algorithms

Object Oriented Design
- Classes are extracted during the object-oriented analysis workflow, and
  - Designed during the design workflow

Workflow (tasks)
Requirements
Analysis
Design
Implementation
Testing

Design workflow

General Design
- Refine the Class Diagram
- Structure system with Subsystems, Interfaces, Classes
- Define subsystems dependencies
- Capture major interfaces between subsystems
- Assign responsibilities to new design classes
- Describe realization of Use Cases
- Assign visibility to class attributes
- Design Databases and needed Data Structures
- Define Methods signature
- Develop state diagram for relevant design classes
- Use Interaction Diagram to distribute behavior among classes
- Use Design Patterns for parts of the system

Workflow (tasks)
Requirements
Analysis
Design
Implementation
Testing

Design workflow

- Architectureal Design
- Identify Design Mechanisms
  - Refine Analysis based on implementation environment
  - Characterize needs for specific mechanisms (inter-process communication, real-time computation, access to legacy system, persistence, ...)
  - Assess existing implementation mechanisms
- Identify Design Classes and Subsystems
  - A Subsystem is a special kind of Package which has behavioral semantics (realizes one or more interfaces)
  - Refine analysis classes
  - Group classes into Packages
  - Identify Subsystems when analysis classes are complex
    - Look for strong interactions between classes
    - Try to organize the UI classes into a subsystem
    - Separate functionality used by different actors in different subsystems
    - Separate subsystems based on the distribution needs
  - Identify Interfaces of the subsystems

Workflow (tasks)
Requirements
Analysis
Design
Implementation
Testing

Implementation workflow

- The aim of the implementation workflow is to implement the target software product in the selected implementation language

Workflow (tasks)
Requirements
Analysis
Design
Implementation
Testing

Implementation workflow

- Distribute the system by mapping executable components onto nodes in the deployment model
- Implement Design Classes and subsystems through packaging mechanism:
  - package in Java, Project in VB, files directory in C++
- Acquire external components realizing needed interfaces
- Unit test the components
- Integrate via builds

Workflow (tasks)
Requirements
Analysis
Design
Implementation
Testing

Testing workflow

- Carried out in parallel with other workflows
- Primary purpose
  - To increase the quality of the evolving system
- The test workflow is the responsibility of
  - Every developer and maintainer
  - Quality assurance group

Workflow (tasks)
Requirements
Analysis
Design
Implementation
Testing

Testing workflow

- Develop set of test cases
  - many for each Use Case
  - each test case will verify one scenario of the use case
  - based on Sequence Diagram
- Develop test procedures specifying how to perform test cases
- Develop test component that automates test procedures

Workflow (tasks)
Requirements
Analysis
Design
Implementation
Testing

Deployment Workflow

- Activities include
  - Software packaging
  - Distribution
  - Installation
  - Beta testing

Deployment workflow

- Producing the Software

Output of implementation is tested executables.
Must be associated with other artifacts to constitute a complete product:
Installation scripts
User documentation
Configuration data
Additional programs for migration: data conversion.
In some cases:
different executables needed for different user configurations
different sets of artifacts needed for different classes of users:
new users versus existing users,
variants by country or language

Deployment workflow

- For distributed software, different sets may have to be produced for different computing nodes in the network Packaging the Software
- Distributing the Software
- Installing the Software
- Migration
- Providing Help and Assistance to Users
- Acceptance

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

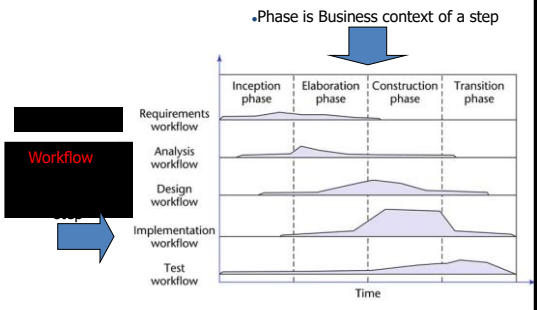
Post delivery maintenance

- Post delivery maintenance is an essential component of software development
  - More money is spent on post delivery 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 post delivery maintenance
  - Regression testing
- All previous test cases (and their expected outcomes) need to be retained

Retirement

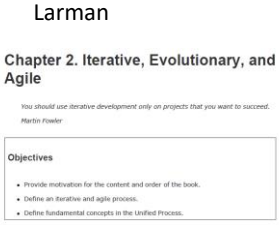
- Software is can be made unmaintainable because
  - A drastic change in design has occurred
  - The product must be implemented on a totally new hardware/operating system
  - Documentation is missing or inaccurate
  - Hardware is to be changed—it may be cheaper to rewrite the software from scratch than to modify it
- These are instances of maintenance (rewriting of existing software)
- True retirement is a rare event
- It occurs when the client organization no longer needs the functionality provided by the product

The Phases/Workflows of the Unified Process



Book Reference and Reading

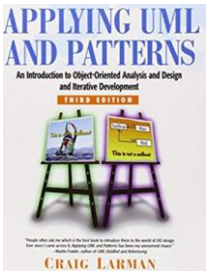
- Chapter 2 – Applying UML & Patterns – Craig Larman


Chapter 2. Iterative, Evolutionary, and Agile

You should use iterative development only on projects that you want to succeed.
Martin Fowler

Objectives

- Provide motivation for the content and order of the book.
- Define an iterative and agile process.
- Define fundamental concepts in the Unified Process.



END OF TOPIC 16

-COMING UP!!!!!!

-4+1 Model view