COMPUTER SCIENCE AND INFORMATION TECHNOLOGY RMIT University

Software Engineering Project Management

Chapter 5: Project Management Methodologies

Course Materials

Online Course Material

Please select a subtopic to view its contents.

Introduction

Waterfall

RUP

Agile

Additional Materials

There are no additional materials available at this time.

COMPUTER SCIENCE AND INFORMATION TECHNOLOGY RMIT University

Software Engineering Project Management

Chapter 5: Project Management Methodologies

Up | Site Map

Introduction

Project Methodologies

- A software development methodology or system development methodology in software engineering is a framework that is used to structure, plan, and control the process of developing an information system.
- Why use one?
  - The tried and tested framework
  - Makes the whole process easier and streamlined
  - Rich documentation available
  - Various tools/software s available
  - Etc.
- Do not use one because a fancy book or a self-obsessed Internet PM-guru recommends it
- Methodologies are no silver bullets
- Their use or over use often leads to obsession over process and the PM looses sight of the project (called methodology-fixation syndrome)
- Methodologies set tone of everything- people, processes, rules and tone of rules. So be very careful how to apply whatever methodology you use
- Don't inflict it on the team
- Use of a particular methodology should never be the sole reason for a project making.

Methodologies and Changing Perspectives

- 1970s
  - Structured programming since 1969
- 1980s
  - Structured Systems Analysis and Design Methodology (SSADM) from 1980 onwards
- 1990s
  - Object-oriented programming (OOP) has been developed since the early 1960s, and developed as the dominant programming methodology during the mid-1990s.
  - Rapid application development (RAD) since 1991.
  - Team software process developed by Watts Humphrey at the SEI
  - Scrum (development), since the late 1990s
- 2000s
  - Extreme Programming since 1999
  - Rational Unified Process (RUP) since 2003.
  - Agile Unified Process (AUP) since 2005 by Scott Ambler
  - Lean

Up | Site Map

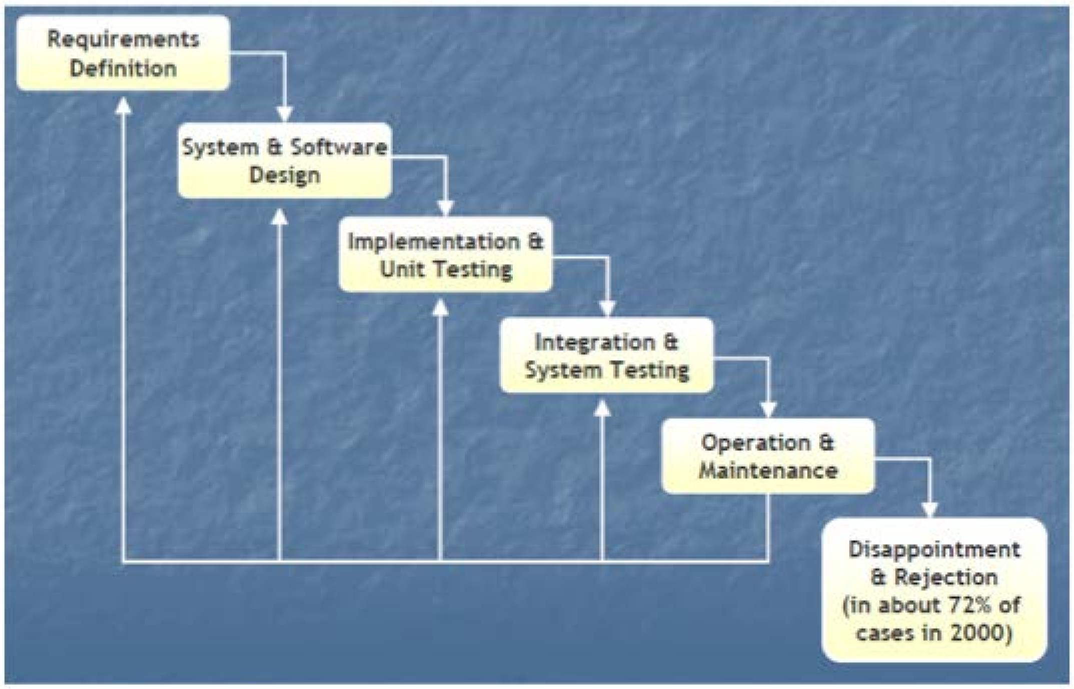
COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
RMIT University

Software Engineering Project Management

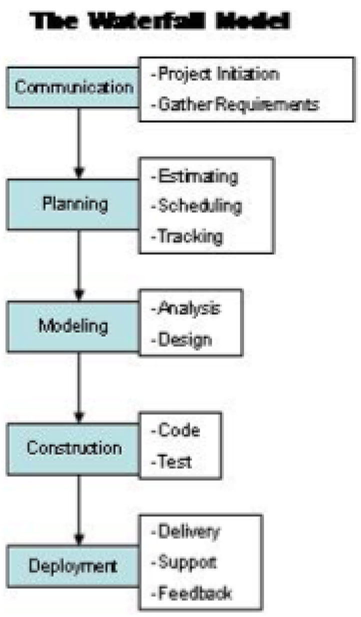
Chapter 5: Project Management Methodologies

Waterfall

Traditional Waterfall


Requirements Definition
System & Software Design
Implementation & Unit Testing
Integration & System Testing
Operation & Maintenance
Disappointment & Rejection (in about 72% of cases in 2000)

Waterfall - Another View



Traditional SDLC

- Linear sequential model
    - Feasibility analysis - preliminary analysis
    - Evaluation analysis
    - System Functional and Non Functional Analysis
    - Design (UI followed by Database)
    - Development
    - Testing
    - Implementation
    - Maintenance / support
- Use of Computer Aided Software Engineering (CASE) tools 1990s
- Size of Project
- When Requirements need to be locked
- Development only starts after Planning
- Output of one step acts as input of next step
- Still widely used
- Separate and distinct phases of specification and development
- Separate and distinct use/application of resources

However...

- It is not flexible to readily incorporate changes.
- Inappropriate for version wise release method.
- The failure / success cant be decided till too late.
- Errors found in later phases difficult to correct
- Many times the customer does not know exactly what the requirements are forcing the team to make their best decisions on what would satisfy the customer
- Blocking States when a part of development is held back - depends on another part that isn't finished yet
- Assume all aspects can be defined prior to the start of work
- Basis for subsequent SDLCs and methodologies

Analysis - Requirements

- Eliciting (vs. gathering) information - what the customer needs (vs. wants)
- understanding the customer's business context and constraints
- functions the product must perform
- performance levels
- external systems it must be compatible with
- Techniques used : customer interviews, use cases, and "shopping lists" of software features
- OUTPUT - formal requirements specification

Design

- defining the hardware and software architecture
- satisfy specified requirements
- specifying performance and security parameters (NFRs)
- strategies to deal with issues such as exception handling, resource management and interface connectivity
- navigation and accessibility
- OUTPUT - one or more design specifications

Implementation

- constructing the product as per the design
- built according to a pre-defined coding standard and debugged
- tested and integrated
- satisfy the system architecture requirements

- OUTPUT - one or more working product components

Testing

- individual components and the integrated whole are methodically verified
- fully meet the requirements
- Three types of testing:
  - unit testing
  - system testing
  - acceptance testing
- And black box and white box
  - OUTPUT...
  - Defects
  - Logged
  - User manual

Installation / Maintenance

- Installation and use at the customer site
- Delivery may take place via the Internet or physical media
- formal revision number
- Maintenance: modifications to the system
- due to change requests initiated by the customer
- defects uncovered during live use of the system
- OUTPUT - "maintenance release"

Assumptions underlying Waterfall model

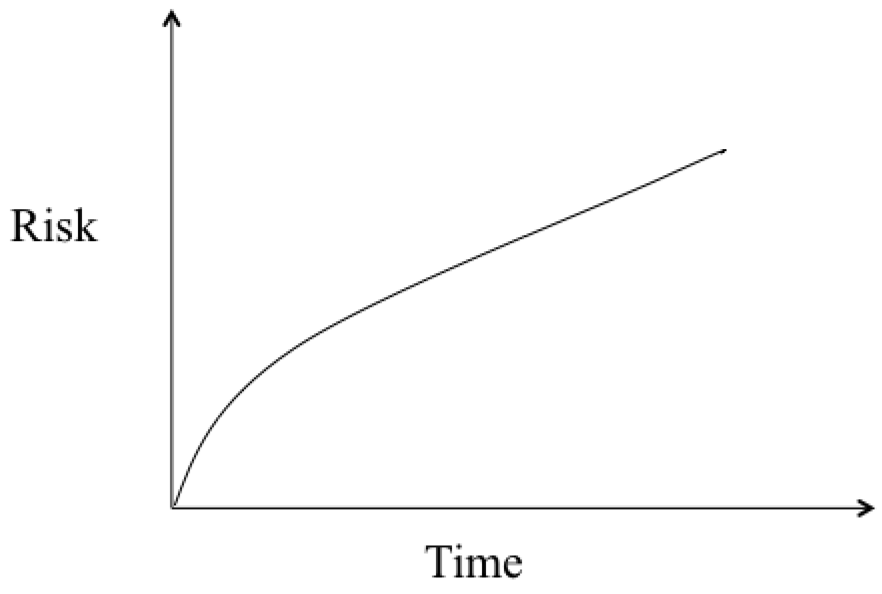
- There exists a reasonably well-defined set of requirements if we only take the time to understand them.
- During the development process, changes to requirements will be small enough that we can manage them without substantially rethinking or revising our plans.
- System integration is an appropriate and necessary process, and we can reasonably predict how it will go based upon architecture and planning.
- Software innovation and the research and development that is required to create a significant new software application can be done on a predictable schedule

Need to use another methodology

- Most of the assumptions stated on slide 16 create issues
- Why?
- In conclusion-
  - Projects can fail to deliver the application as intended to the customer in the predicted time frame.
  - Waterfall model solution is somewhat off the mark
  - Reworking may be a big issue
  - Risk management may be very hard to integrate
  - the list goes on..

Need to use another methodology

- ***Risk vs Time***



Up | Site Map

COMPUTER SCIENCE AND INFORMATION TECHNOLOGY RMIT University

Software Engineering Project Management

Chapter 5: Project Management Methodologies

Up | Site Map

Rational Unified Process

RUP

- The Rational Unified Process activities create and maintain models. Rather than focusing on the production of large amount of paper documents.
- The Rational Unified Process is a guide for how to effectively use the Unified Modeling Language (UML).
- captures many of the best practices Compared to the traditional waterfall process, the iterative process has the following advantages:
    - Risks are mitigated earlier
    - Change is more manageable
    - Higher level of reuse
    - The project team can learn along the way
    - Better overall quality

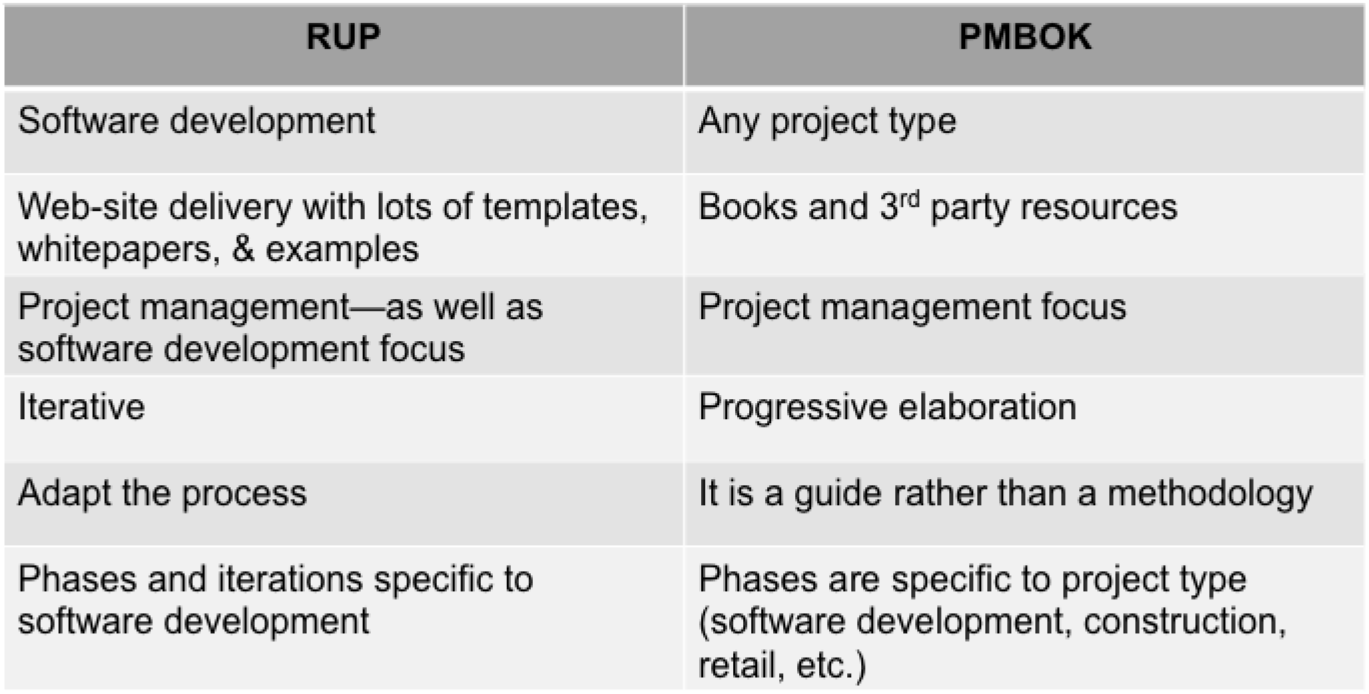
It is not a guide / book

- RUP is not a book, it is a development method developed and published once and for all in paper form. The RUP is designed, developed, delivered, and maintained like any software tool.
- The RUP shares many characteristics with software products

RUP and software development process

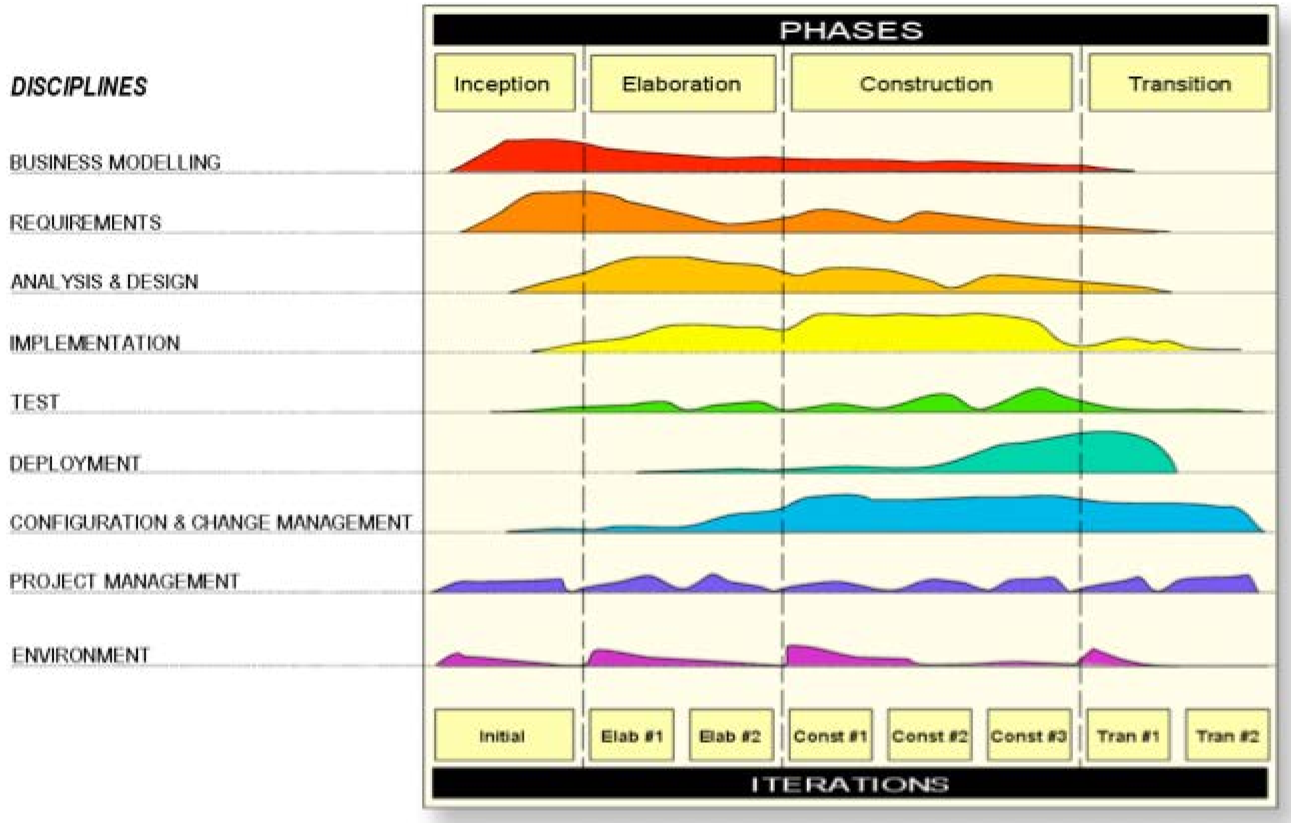
- An iterative and incremental software development process
- Originally developed by Rational Software, which was acquired by IBM in February 2003 Complete Life Cycle Software Engineering Process
- Iterative and Incremental Approach
- Use Case Driven approach
- Architecture Centric

How does RUP compare to the PMBOK?



RUP	PMBOK
Software development	Any project type
Web-site delivery with lots of templates, whitepapers, & examples	Books and 3rd party resources
Project management—as well as software development focus	Project management focus
Iterative	Progressive elaboration
Adapt the process	It is a guide rather than a methodology
Phases and iterations specific to software development	Phases are specific to project type (software development, construction, retail, etc.)

Rational Unified Process Architecture



RUP (objective and output)

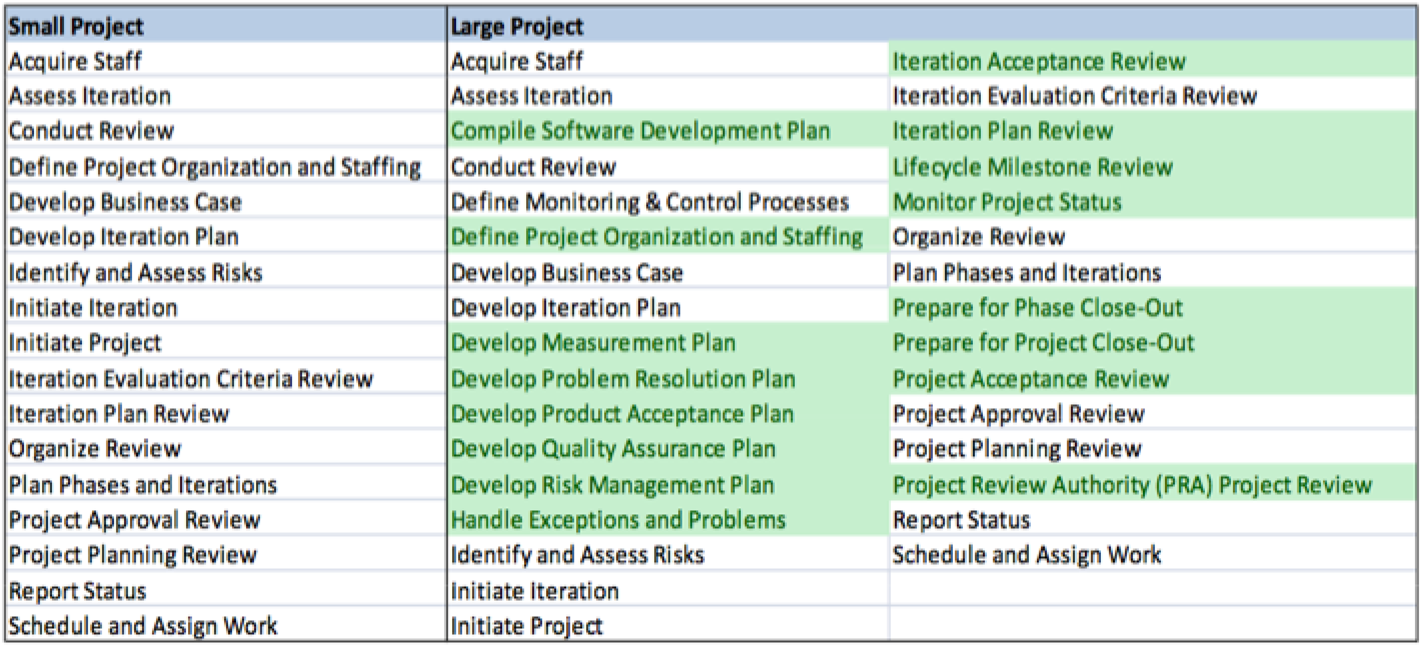
1. INCEPTION
    - Objective

        - Establish Project Scope
        - Use Case Identification
        - Estimate Cost
        - Prioritize Risk
    - Output
        - A vision document
        - Use cases
        - An initial project glossary
        - An initial business case
        - An initial risk assessment
        - A project plan
2. ELABORATION
    - Objective
        - Define And Validate System Architecture .
        - Baseline Vision.
        - Reliable Estimate of cost.
    - Output
        - Use Case Model
        - Software Architecture Description
        - Revise Risk Table
        - Preliminary User Manual
3. CONSTRUCTION
    - Objectives
        - Optimize Resource Usage
        - Avoid unnecessary changes
        - Produce quality code
    - Output
        - Components
        - Classes
        - Objects
4. TRANSITION
    - Objectives
        - From development to deployment
        - Training users
        - Beta Testing
    - Output
        - Product
        - User Manual

RUP PM tasks

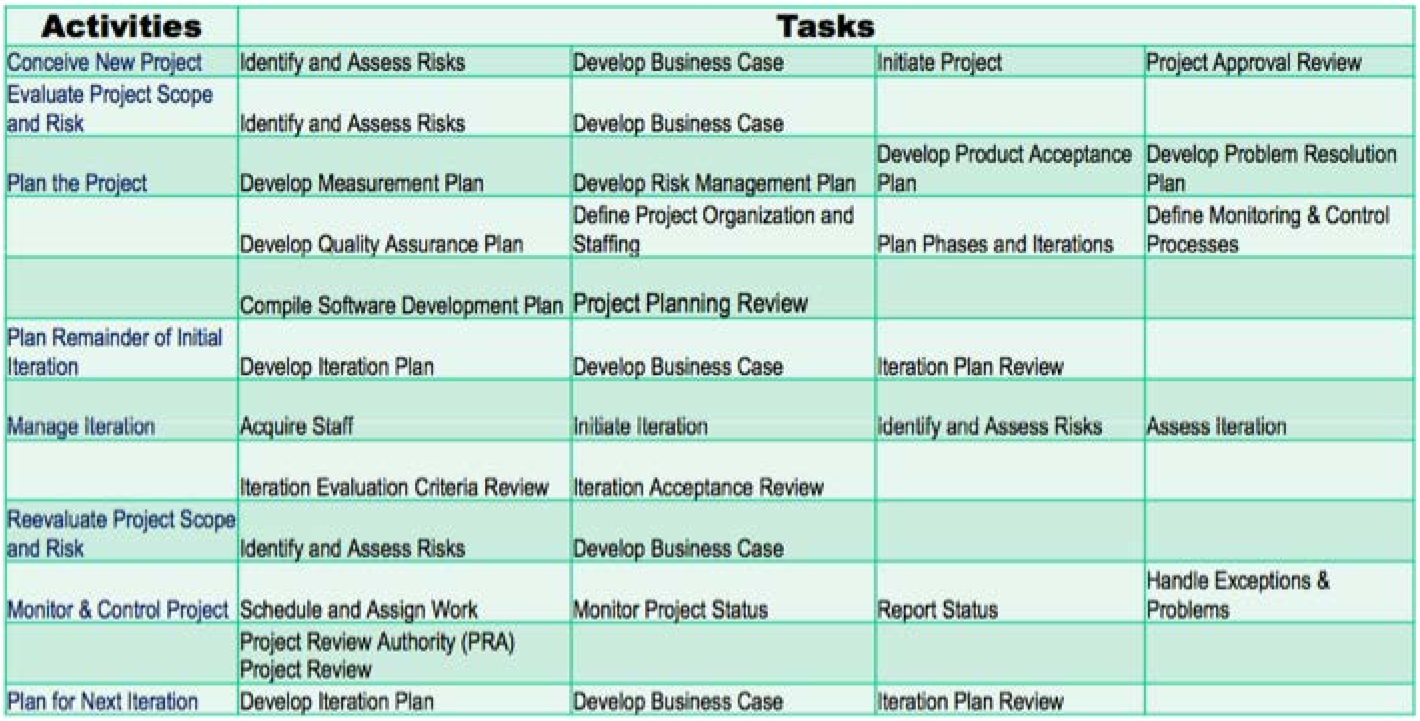
- Small projects have 17 RUP tasks.
- Large projects have 32 RUP tasks.
- PMBOK has 42 processes. Not all PMBOK processes map to RUP tasks.

RUP PM tasks

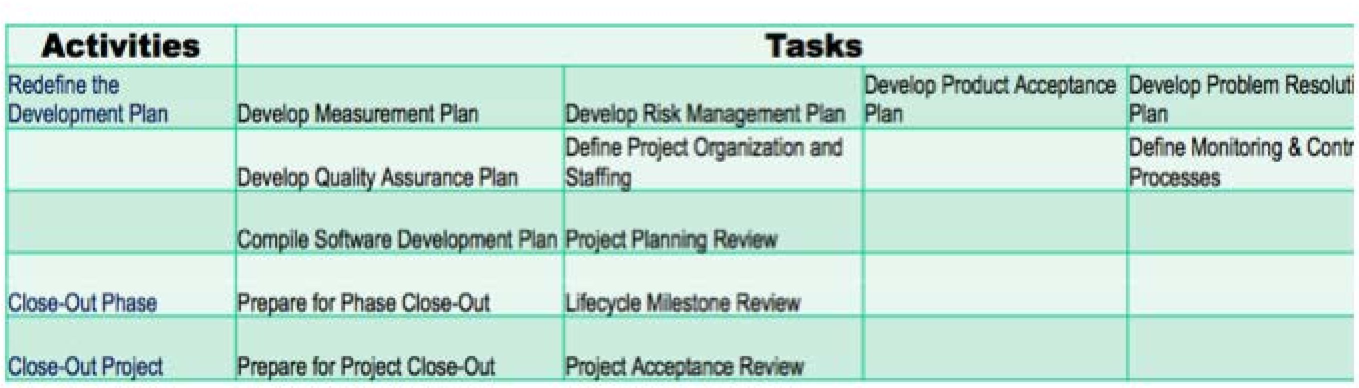


Small Project	Large Project	
Acquire Staff	Acquire Staff	Iteration Acceptance Review
Assess Iteration	Assess Iteration	Iteration Evaluation Criteria Review
Conduct Review	Compile Software Development Plan	Iteration Plan Review
Define Project Organization and Staffing	Conduct Review	Lifecycle Milestone Review
Develop Business Case	Define Monitoring & Control Processes	Monitor Project Status
Develop Iteration Plan	Define Project Organization and Staffing	Organize Review
Identify and Assess Risks	Develop Business Case	Plan Phases and Iterations
Initiate Iteration	Develop Iteration Plan	Prepare for Phase Close-Out
Initiate Project	Develop Measurement Plan	Prepare for Project Close-Out
Iteration Evaluation Criteria Review	Develop Problem Resolution Plan	Project Acceptance Review
Iteration Plan Review	Develop Product Acceptance Plan	Project Approval Review
Organize Review	Develop Quality Assurance Plan	Project Planning Review
Plan Phases and Iterations	Develop Risk Management Plan	Project Review Authority (PRA) Project Review
Project Approval Review	Handle Exceptions and Problems	Report Status
Project Planning Review	Identify and Assess Risks	Schedule and Assign Work
Report Status	Initiate Iteration	
Schedule and Assign Work	Initiate Project	

RUP Activities and tasks for a larger project



Activities	Tasks			
Conceive New Project	Identify and Assess Risks	Develop Business Case	Initiate Project	Project Approval Review
Evaluate Project Scope and Risk	Identify and Assess Risks	Develop Business Case		
Plan the Project	Develop Measurement Plan	Develop Risk Management Plan	Develop Product Acceptance Plan	Develop Problem Resolution Plan
	Develop Quality Assurance Plan	Define Project Organization and Staffing	Plan Phases and Iterations	Define Monitoring & Control Processes
	Compile Software Development Plan	Project Planning Review		
Plan Remainder of Initial Iteration	Develop Iteration Plan	Develop Business Case	Iteration Plan Review	
Manage Iteration	Acquire Staff	Initiate Iteration	Identify and Assess Risks	Assess Iteration
	Iteration Evaluation Criteria Review	Iteration Acceptance Review		
Reevaluate Project Scope and Risk	Identify and Assess Risks	Develop Business Case		
Monitor & Control Project	Schedule and Assign Work	Monitor Project Status	Report Status	Handle Exceptions & Problems
	Project Review Authority (PRA) Project Review			
Plan for Next Iteration	Develop Iteration Plan	Develop Business Case	Iteration Plan Review	



Activities	Tasks			
Redefine the Development Plan	Develop Measurement Plan	Develop Risk Management Plan	Develop Product Acceptance Plan	Develop Problem Resoluti Plan
	Develop Quality Assurance Plan	Define Project Organization and Staffing		Define Monitoring & Contr Processes
	Compile Software Development Plan	Project Planning Review		
Close-Out Phase	Prepare for Phase Close-Out	Lifecycle Milestone Review		
Close-Out Project	Prepare for Project Close-Out	Project Acceptance Review		

Up | Site Map

COMPUTER SCIENCE AND INFORMATION TECHNOLOGY RMIT University

Software Engineering Project Management

Chapter 5: Project Management Methodologies

Agile

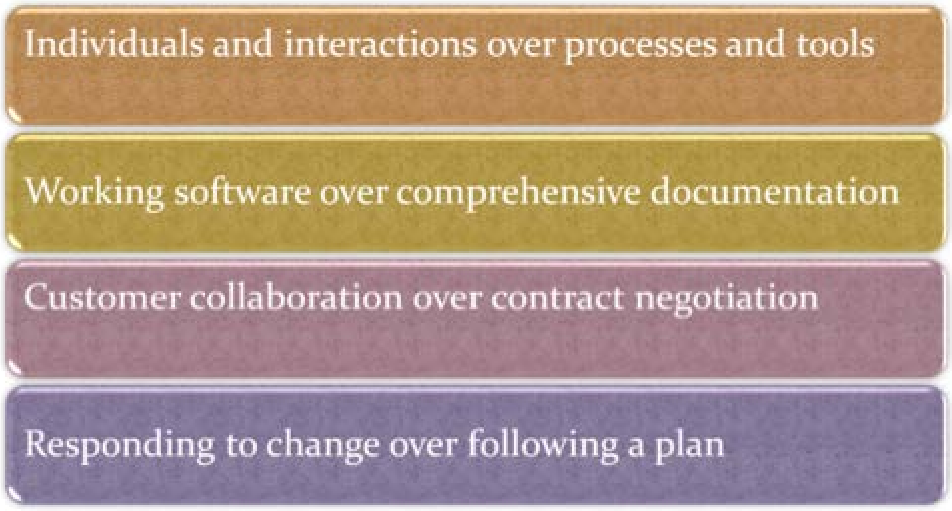
Agile

Definition:

An iterative and incremental (evolutionary) approach to software development which is performed in a highly collaborative manner by self-organizing teams within an effective governance framework with "just enough" ceremony that produces high quality software in a cost effective and timely manner which meets the changing needs of it's stakeholders.

- Best suited for where requirements keep changing
- Based on iterative and incremental approach
- Deliver working software quickly
- Customer involved in the development process
- System delivery in small increments
- User can use the system from the beginning
- Rapid application development environment

Agile Manifesto



Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

Agile Principles

- Simplicity
- Highest priority : customer satisfaction
- Welcome changing requirements
- Delivery of working software frequently
- Business people and developers work together daily.
- Motivate individual :provide environment and support : trust
- face-to-face conversation
- Working software is the primary measure of progress
- Attention to technical excellence and good design
- The best architectures, requirements, and designs emerge from self-organizing teams.

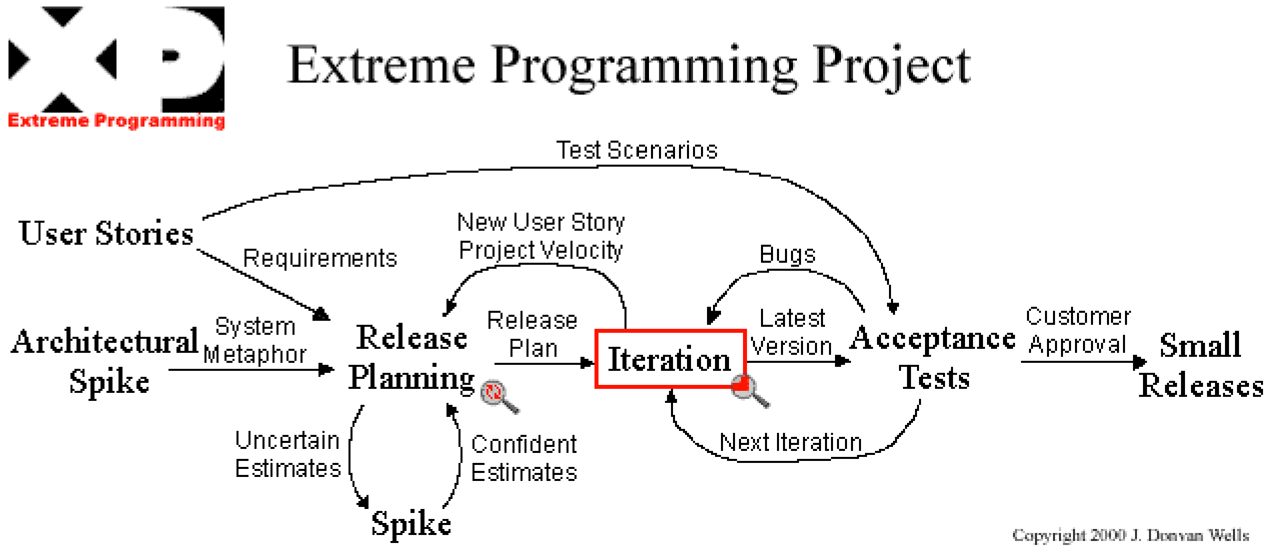
At regular intervals : Team checks how to become more effective

Existing Agile Methods

- Extreme Programming ("XP")
- Agile Unified Process (AUP)
- SCRUM

XP

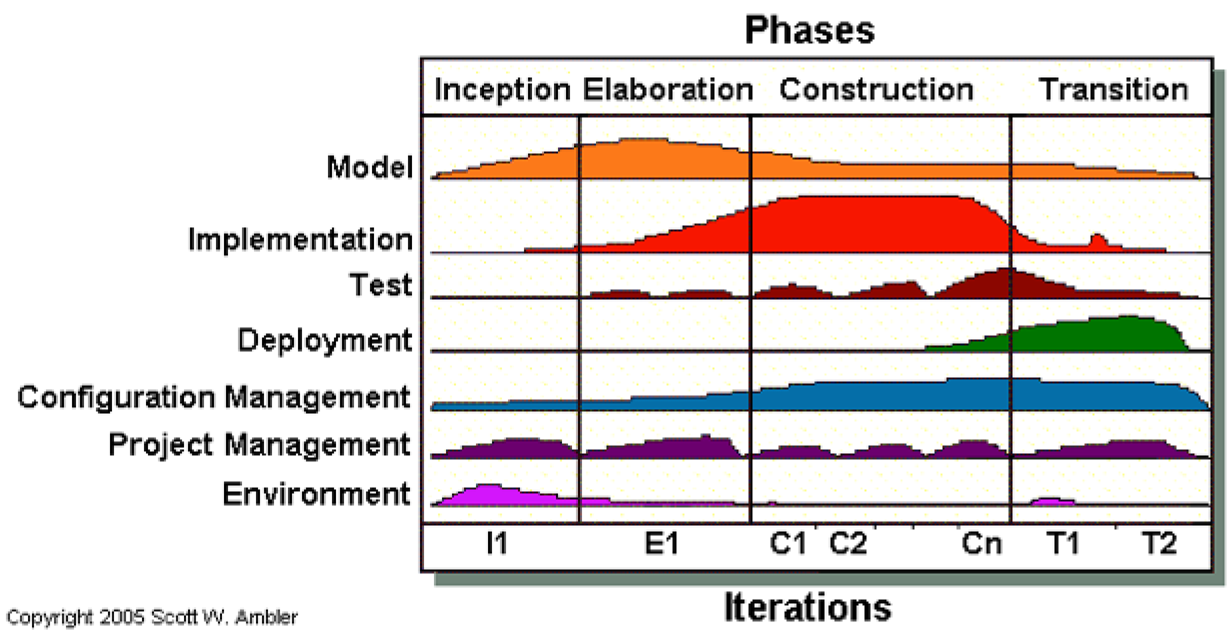
- Extreme Programming improves a software project in five essential ways; communication, simplicity, feedback, respect, and courage.
- Extreme Programmers constantly communicate with their customers and fellow programmers. They keep their design simple and clean.
- They get feedback by testing their software starting on day one. They deliver the system to the customers as early as possible and implement changes as suggested.



Is XP dead?

- NO!
- However, on its own is definitely no longer popular
- New Agile methodologies have forayed
- Teams do use XP practices as a part and parcel of the complete process package the team adopts , so while on the face of it they say it's Scrum but in practice they use a bit of Scrum, bit of Kanban, bit of XP etc.

AUP



AUP vs. RUP

- The first thing that you'll notice is that the disciplines have changed.
- First, the Model discipline encompasses the RUP's Business Modeling, Requirements, and Analysis & Design disciplines.
- Model is an important part of the AUP, as you can see, but it doesn't dominate the process -- you want to stay agile by creating models and documents which are just barely good enough.
- Second, the Configuration and Change Management discipline is now the Configuration Management discipline. In agile development your change management activities are typically part of your requirements management efforts, which is part of the Model discipline.

AUP Philosophy

- Your staff knows what they're doing. People aren't going to read detailed process documentation, but they will want some high-level guidance and/or training from time to time. The AUP product provides links to many of the details, if you're interested, but doesn't force them upon you.
- Simplicity. Everything is described concisely using a handful of pages, not thousands of them.
- Agility. The Agile UP conforms to the values and principles of the Agile Alliance.
- Focus on high-value activities. The focus is on the activities which actually count, not every possible thing that could happen to you on a project.
- Tool independence. You can use any toolset that you want with the Agile UP. My suggestion is that you use the tools which are best suited for the job, which are often simple tools or even open source tools.
- You'll want to tailor this product to meet your own needs. The AUP product is easily tailorable via any common HTML editing tool. You don't need to purchase a special tool, or take a course, to tailor the AUP.

SCRUM

- Best software development process (?)
- Product Backlog
- Team Roles
- Product owner – Customer /Executive
- Scrum Master
- Developer
- Tester
- Sprints

- Burndown charts

- A user story is a basic unit of work in an agile project
  - Business Requirement
- Product Backlog
  - All the feature that needs to be implemented in the software or product
  - All the business requirements

Structure of a basic unit of SCRUM

- Themes, Epics and User stories
- A user story is a basic unit of work in an agile project
- Approach
  1. "As a..."
  2. "I want to..."
  3. "Under these conditions..."
  4. "So that..."
  5. Optional
     a. "Data inputs"
     b. "Data Outputs"
     c. "With volumes and frequency of"
  6. Acceptance Criteria
     a. "Given"
     b. "When"
     c. "Then"

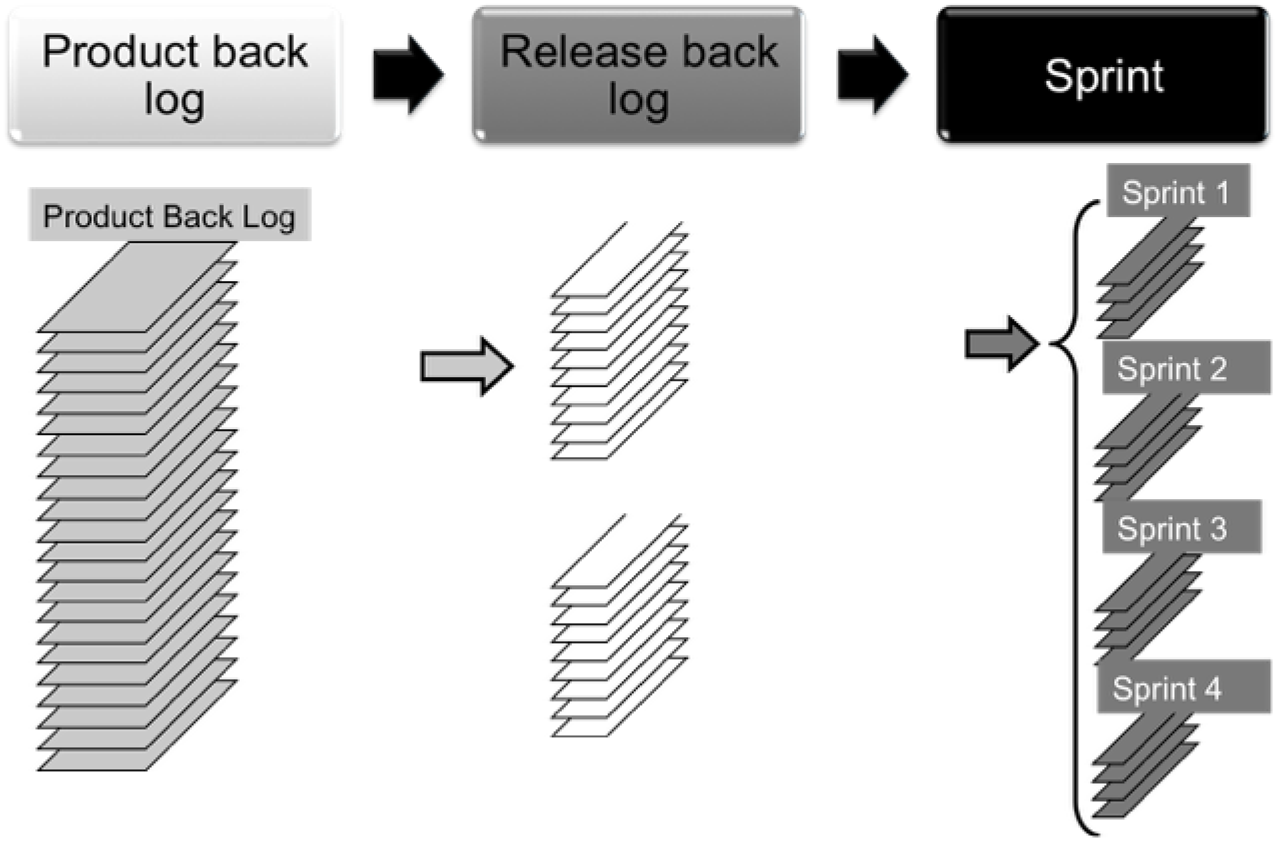
Release Backlog

- All the features that need to be implemented in the software or product
- Typically 2 – 4 months
-

SCRUM Sprint

- Sprint is a collection of stories the team commits to delivering in a fixed period of time
- Typically 2-4 weeks
- Every iteration, the team commits to delivering the stories chosen by the customer
-

SCRUM Overview



Lesson Number 5

- Don't do agile like this!!
- Chet Rong way to do Agile

Acknowledgement: YouTube video's copyright is held by the YouTube website and the owner, it has been cited for educational-purposes.

References

- http://www.mydeskdrawer.com/projectmanagement/gantt.html
- http://en.wikipedia.org/wiki/Project_plan
- http://en.wikipedia.org/wiki/Schedule_(project_management)
- http://www.netmba.com/operations/project/pert/
- http://office.microsoft.com/en-us/help/HA011361531033.aspx#Step%201
- http://www.tensteppb.com/5.3.01.1TSCreateWBSProcess.htm
- http://www.projectlearning.net/pdf/E2_1.pdf
- http://blogs.msdn.com/project/archive/2008/07/29/back-to-basics-understanding-task-dependencies.aspx

Suggested Reading

- None in the prescribed textbook, but here are two very relevant and good readings-
- RUP
- Search for this IBM paper – 'Software Project Management - A Mapping between RUP and the PMBOK' .

Up | Site Map