

# Software Life Cycle

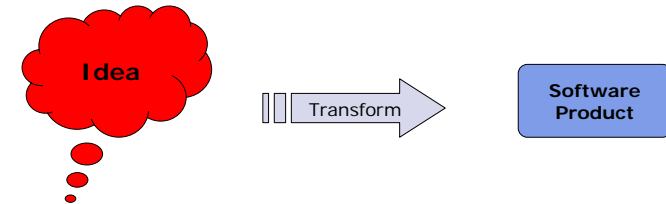
Lecture #17  
Software Engineering and  
Project Management

Instructed by Steven Choy on Mar 5, 2007



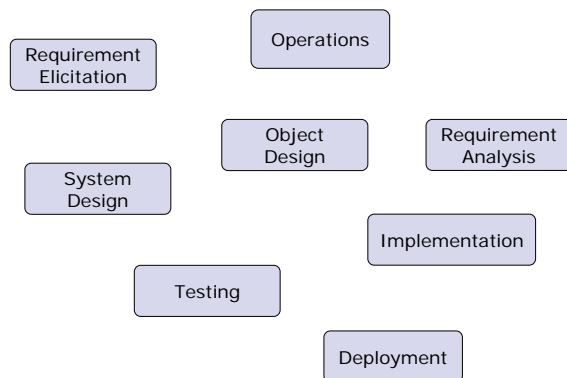
1

# The Goal of Software Development



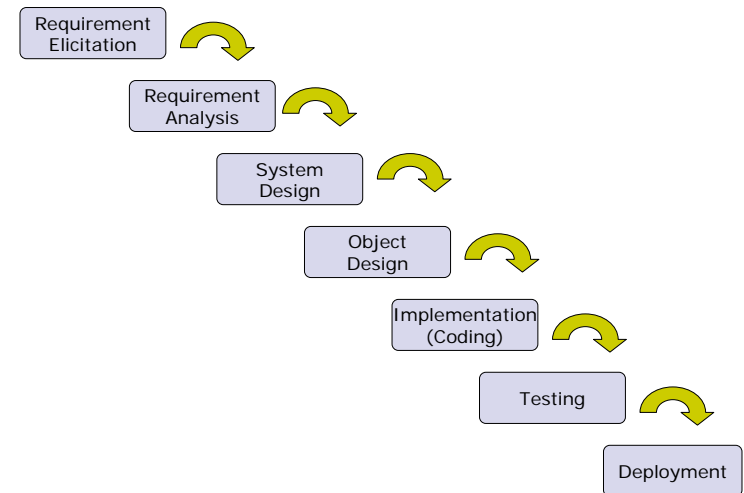
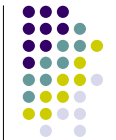
2

# Software Development Activities



3

# We can arrange like this...



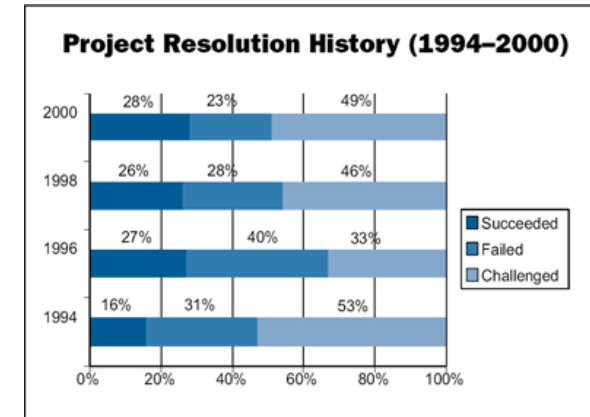
4

## What we will discuss today?

- Software Development Life Cycle (SDLC)
  - Waterfall Model
  - Spiral Model
  - Rational Unified Process (RUP)
- Process Maturity
  - Capability Maturity Model (CMM)



## SDLC: Why?



Project success rates are rising. This chart depicts the resolution of the 30,000 applications projects in large, medium, and small cross-industry US companies tested by The Standish Group since 1994.

Source: standishgroup.com

## SDLC: Why?

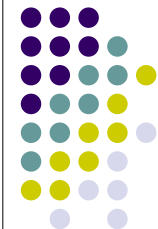
The CHAOS Ten

Executive Support	18
User Involvement	16
Experienced Project Manager	14
Clear Business Objectives	12
Minimized Scope	10
Standard Software Infrastructure	8
Firm Basic Requirements	6
Formal Methodology	6
Reliable Estimates	5
Other	5

*Each factor has been weighted according to its influence on a project's success. The more points, the lower the project risk.*

Source: standishgroup.com

## Software Life Cycle Model



# Software Life Cycle Model



- A general model of the software management process, including all activities and work products required to develop a software
- Model types:
  - Activity-centered: Focus on the activities of software development
  - Entity-centered: Focus on work products created by the activities
- Sequential, iterative, or incremental?

# Typical Software Life Cycle Models



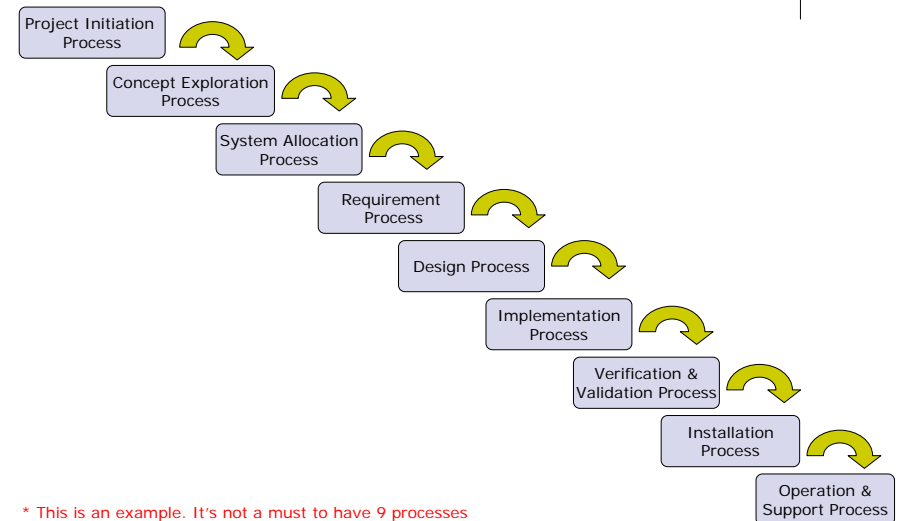
- Waterfall Model
- V-Model
- Spiral Model

# Waterfall Model



- An activity-centred life cycle model
- Prescribes sequential executions of development processes and management processes

# Waterfall Model Processes



\* This is an example. It's not a must to have 9 processes



## Characteristics of Waterfall Model

- Complete each activities one by one and never “look back”
  - Freeze requirement before design
  - Avoid coding until you have a detailed design
  - Complete unit testing before integration
- Provides a simple view of software development that measures progress by the number of tasks completed
  - But does the progress really reflect where we are?

13

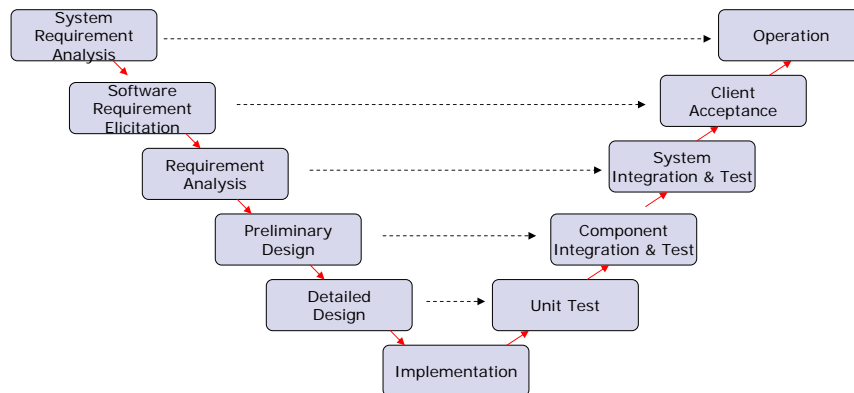


## V-Model

- Also a sequential activity-centred model
- A variant of waterfall model
- Goal: Associates development activities with verification activities

14

## V-Model Illustrated



15

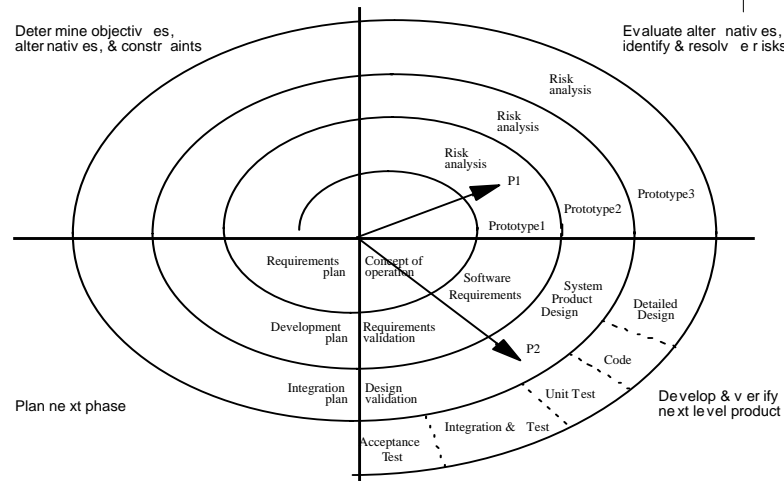


## Spiral Model

- Also an activity-centred model
- But spiral model is designed to address the weaknesses in waterfall model, in particular, **to accommodate infrequent change** during software development

16

## Spiral Model Illustrated



17

## Activities in Spiral Model



- 1<sup>st</sup> Phrase (Upper-left quadrant)
  - Identify objectives and define constraints and
- 2<sup>nd</sup> Phrase (Upper-right quadrant)
  - Identify and resolve risks
- 3<sup>rd</sup> Phrase (Lower-right quadrant)
  - Develop and verify next-level product
- 4<sup>th</sup> Phrase (Lower-left quadrant)
  - Plan next phrases

18

## Unified Process



- Better known as Rational Unified Process (RUP)
- Developed by Booch, Jacobson and Rumbaugh from Rational Software Corporation (now part of IBM)
- Explicitly designed to support the implementation of best practices
- Characteristics:
  - Use-case Driven
    - Advocates use-case over traditional functional specification
  - Architecture Driven
    - Emphasizes on the use of modeling
  - Iterative

19

## The Six Practices



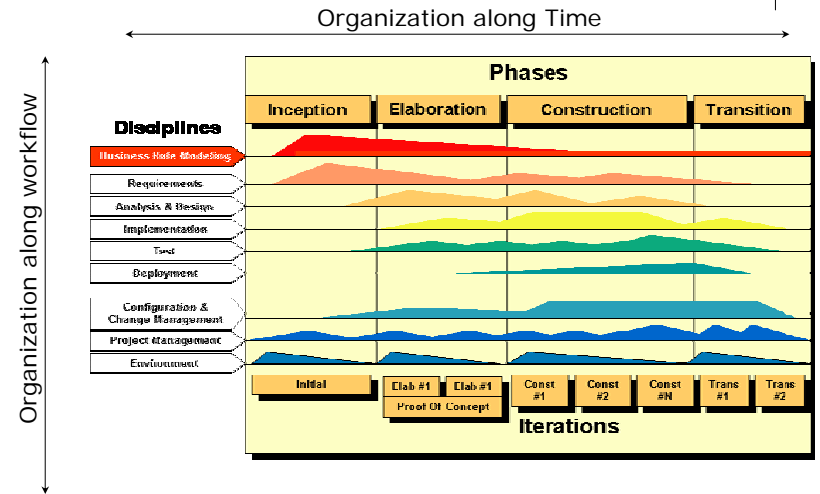
- Develop iteratively
  - It's more practical to divide a complex project into small project and grows it incrementally
  - Allows increase understanding of the problem
  - Each iteration ends with an executable release
  - Better accommodates for requirement, schedule or feature changes
- Manage requirements
  - Describe how to elicit and document requirement, features and constraints
  - Use-case model
- Use component-based architecture
  - Support component-based development
  - Describe how to design a flexible architecture that accommodates changes and promotes reuse

20

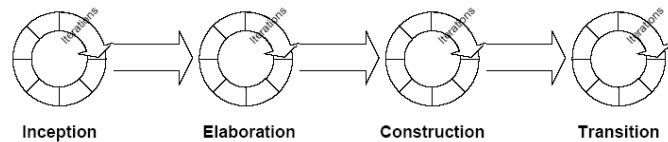
# The Six Practices

- Model visually
  - Describe how to model software visually to capture the structure and behaviour of components
  - Do you remember those UML diagrams?
- Verify quality continuously
  - Quality assessment is built into the process, in all activities and involves all participants
  - Quality is reviewed respect to the requirement
- Manage change
  - Describe how to track, control and monitor changes
  - Changes: code change, requirement change, design change

# Rational Unified Process



# Phases and Iterations



- The software lifecycle is broken in phases
  - Inception
  - Elaboration
  - Construction
  - Transition
- Each phase can be further divided into several iterations

# Inception Phase

- Focus on establishment of business case and where concurrence among all stakeholders on the objectives for the project

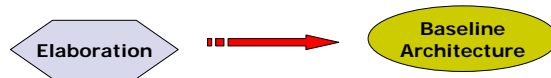


- Outcome:
  - A vision document: core project requirements, key features and associated constraints
  - An initial use case model (10-20% complete)
  - An initial business case
  - An initial risk assessment
  - A project plan
  - Preliminary prototypes

## Elaboration Phase



- Establishes the software architecture that provides a stable foundation for the design and implementation



- Outcome:
  - A use case model (~80% complete)
  - A software architecture description
  - An executable software prototype
  - A revised risk assessment and business case
  - An updated development plan

25

## Construction Phase



- Complete the development of the system based on the baselined architecture in elaboration phase
- Clarifies the remaining requirements



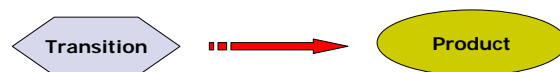
- Outcome:
  - A product ready to put in the hand of end users
  - User manual
  - Release note

26

## Transition Phase



- Deploy the system to user environment for evaluation and testing
- Typically this phase includes several iterations:
  - Beta release
  - General availability release
  - Bug fix or enhancement release



27

## RUP: An Iterative Approach



- Each phrases comprises with one or several iterations
- Each complete iteration results a release
- The final system is grown incrementally from iteration to iteration
- Benefit:
  - Risk is mitigate earlier
  - Learn along the way
  - Change is more manageable
  - Better overall quality

28

# RUP Workflows

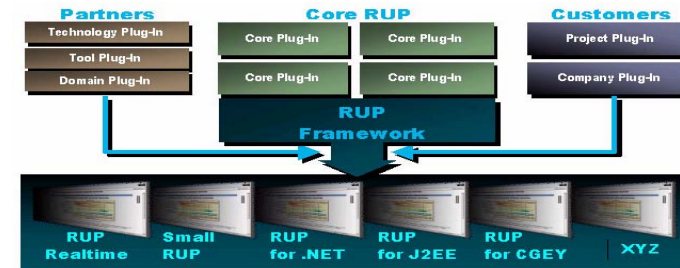


- RUP activities are known as workflows:
  - Business Modeling
  - Requirements
  - Analysis and Design
  - Implementation
  - Test
  - Configuration Management
  - Project Management
  - Environment
  - Deployment
- In an iteration, all the workflows are performed

# RUP is NOT JUST a Process?



- RUP is also a **process framework**
- RUP process framework is an extensible architecture for process definition



# Designing Software Life Cycle Process



None of the existing processes suits my organization's need.

How can I create my own?



## Designing SLCP



- **IEEE 1074** standard is designed to help architect/project manager to design Software Life Cycle Process (SLCP) after selecting a software life cycle model (SLCM)
- Seeks to establish a common framework for developing life cycle models
- Describes a set of activities and processes that are mandatory for the development and maintenance of software

33

## Core Activity Groups



- Project Management
  - Project Initiation
  - Project Planning
  - Project Monitoring & Control
- Pre-development
  - Concept Exploration
  - System Allocation
  - Software Importation
- Development
  - Requirements
  - Design
  - Implementation
- Post-development
  - Installation
  - Operation & Support
  - Maintenance
  - Retirement
- Integral
  - Evaluation
  - Software Configuration Management
  - Documentation Development
  - Training

34

## Process Maturity



35

## Software Process Maturity



- What does maturity mean to you?
  - A "mature" process reduces the risk of project failure and increase the predictability/quality of software project
- A software organization is said to be mature if
  - The development activities are well-defined and documented
  - Products are always delivered on schedule and within budget (Of course, without compromising the quality)
  - The software process is accurately communicated to both existing staffs and new comers, and work activities are carried out according to the planned process

36

## Measuring Process Maturity



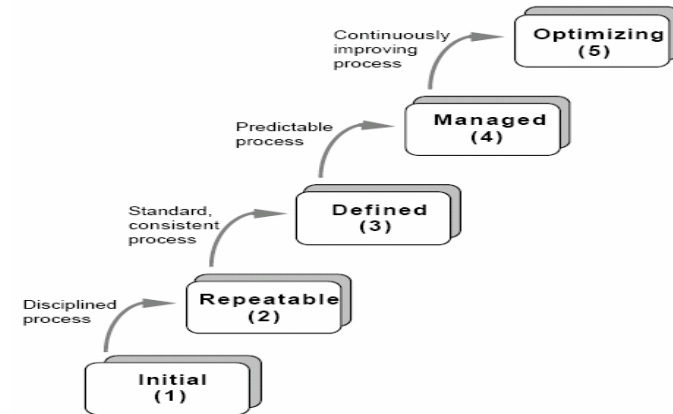
- Capability Maturity Model for Software (SW-CMM / CMM)
- Developed by Software Engineering Institute (SEI) of Carnegie Mellon University (CMU)
- What's CMM?
  - A framework **describing** an evolutionary path from ad-hoc, chaotic process to mature, disciplined software process
  - Establishes a set of criteria **describing** the characteristic of mature software organization
  - Covers practices for project planning, engineering and management

37

## Capability Maturity Level



- CMM classifies maturity into five levels



38

Source: Capability Maturity Model for Software v1.1 (CMU/SEI-93-TR-024)

## Capability Maturity Level



- Initial (Level 1)
  - Ad-hoc development, no formal project plan, no documentation, code & fix style
  - Success depends on individual effort
  - Products are often delivered late and with poor quality
- Repeable (Level 2)
  - Basic project management processes to manage software projects (track cost, schedule)
  - Earlier successes can be repeated
- Defined (Level 3)
  - With established organization-wide standard processes for software development
  - All projects follow the organization-wide processes to develop and maintain software
- Managed (Level 4)
  - Activities are measured quantitatively to provide feedback for better resource management
- Optimizing (Level 5)
  - Process is continuously improved by feedback of quantitative measurement

39

## What does Maturity Level indicate?



- Each level indicates a level of predictability of your project performance
  - **Initial level:** Random, unpredictable performance. Schedule and cost are typically overrun
  - **Repeable level:** Repeable performance from project to project. Plan based on past experience is more realistic
  - **Defined level:** Better performance based on well-defined organization-wide process
  - **Managed level:** Project performance continues to improve based on quantitative understanding of the process
  - **Optimized level:** Project performance continues to improve

40

## Higher Level, Less Bugs



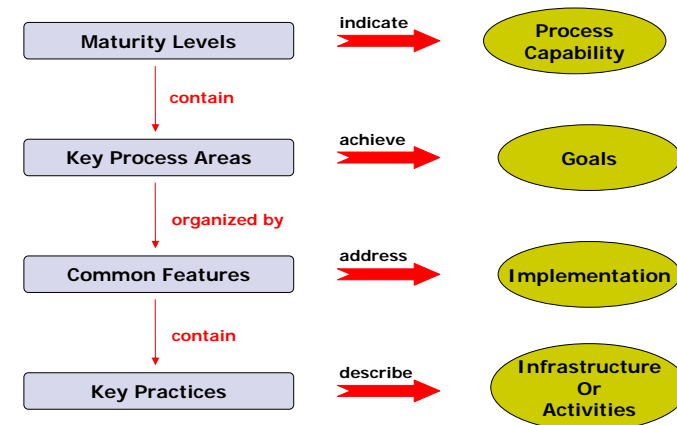
- Higher CMM level correlate with fewer software bugs

CMM Level	Average defects per function point	% Improvement (from previous level)	Cost savings per 100 function points at U.S. labour rates	Cost savings per 100 function points at offshore labour rates
1	0.750	-	-	-
2	0.620	17.33	\$14,560	\$6,240
3	0.475	23.34	\$16,240	\$6,690
4	0.228	52.00	\$27,664	\$11,856
5	0.100	56.00	\$14,336	\$6144

41

Source: Gartner Inc. and the Software Engineering Institute

## CMM Structure



42

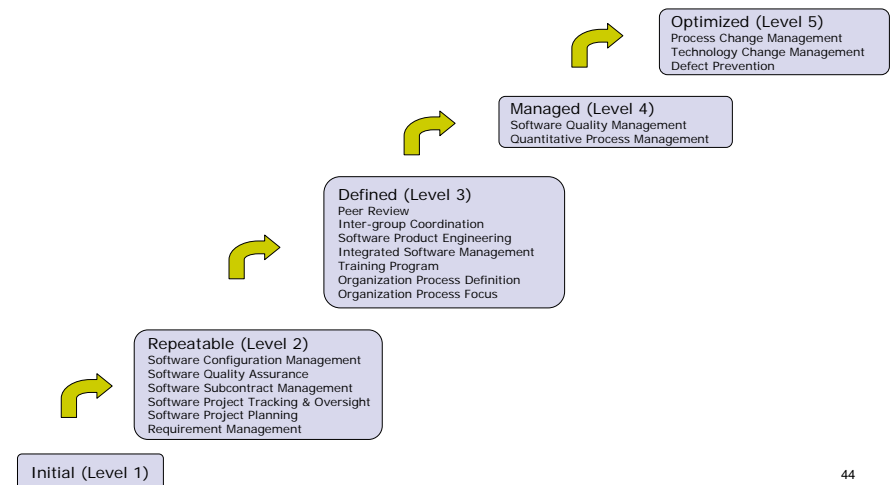
## CMM: Key Process Areas



- Each maturity level is decomposed into several key process areas
- Each key process area identifies a cluster of related activities that, when performed collectively, achieve a set of goals considered important for enhancing process capability.
- To achieve a maturity level, an organization must fulfill all key process areas (KPA) defined for that level

43

## Key Process Areas



44

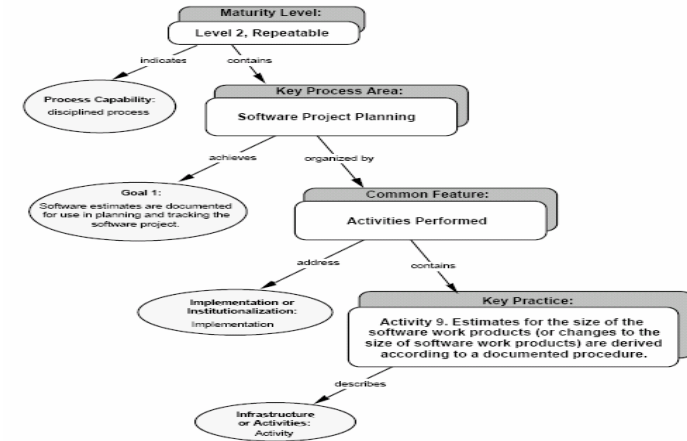
## Key Practices



- Each KPA is described in terms of Key Practices that contribute to satisfying its goals
- The key practices describe the infrastructure and activities that contribute most to the effective implementation of the key process area
- Each Key Practice consists of a single sentence (followed by detailed description), describing **“what” to be done, but not “how”**

45

## Example of Key Practice



46

Source: Capability Maturity Model for Software v1.1 (CMM/SEI-93-TR-024)

## How is CMM used in practice?



- Improve the software development process of your organization
  - CMM has specific value in the area of action planning, implementing action and defining processes
  - Compare existing practices against the goals of key process areas in CMM
- Evaluate outsourcing vendors/contractors
  - CMM maturity level also serves as an indicator of how well the software development process an organization has
  - A vendor with high maturity level is preferred than the lower counterpart
  - So, most outsourcing vendors adopts CMM and tries to achieve level 5 maturity
  - Example: Infosys from India: CMM-level5
  - More can be found at

<http://sas.sei.cmu.edu/pars/pars.aspx>

47

## Who gives me the Appraisal?



- Okay, I know my organization deploys all key process areas for Level-3. Can I say my organization is at CMM Level-3?
- But one thing is exceptional:

**Everyone can say, “I’m at CMM Level-1”**

48

## CMM Assessment



- Team Selection
- Maturity Questionnaire
- Response Analysis
- On-site Visit, with interviews and document review
- Report findings based on the CMM
- Prepare KPA Profile

49

## Benefits of CMM



- Increased the control of project (more predictable in terms of both cost and schedule)
- Predictability of the effect of a software change on project cost or schedule
- Better the development process

50

## Capability Maturity Model Integration



- Abbreviated as CMMI
- An upgrade version of CMM
- Changes:
  - Added new process areas
    - Example: Added measurement and analysis to process areas at level-2
  - Added Modern best practices
  - Split into staged and continuous representation

51

## Acknowledgement

The slides were originally authored by **Simon Ng**.



52